Applications for SAM-2

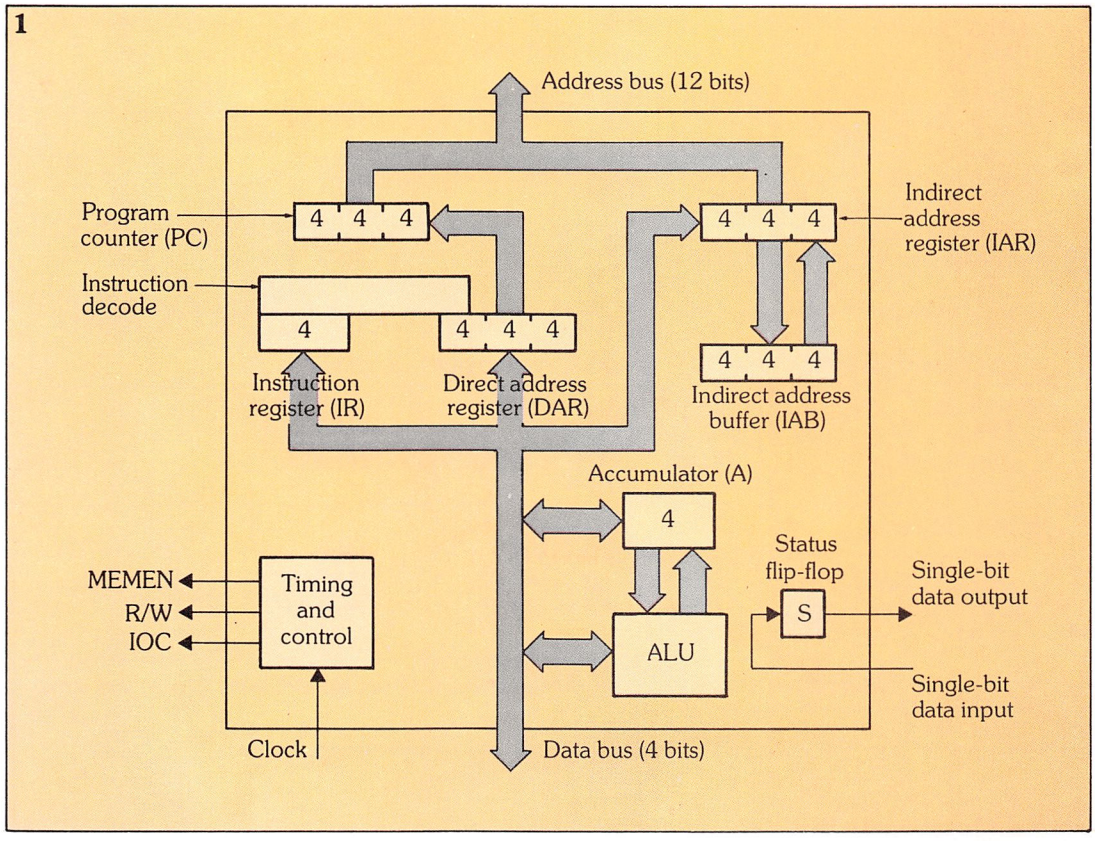
SAM – a typical microprocessor

Having examined basic microprocessor operation, we are now in a position to look at the application of a microprocessor to a specific problem. A fictitious microprocessor called SAM (Simplified Architecture Microprocessor) will illustrate most of the basic features of a typical microprocessor.

Inside a microprocessor

SAM's internal layout is shown in *figure 1* and, although less complex than a real microprocessor, all the essential registers are the same. The 12-bit program counter (PC) and 12-bit address bus enable SAM to address any of 4096 (2^{12}) memory locations. When an instruction addressed by the program counter is received, the instruction register (IR) provides four bits of storage for the instruction operation code (OP code). A 12-bit direct address register (DAR) holds any direct data addresses contained in the instruction. All this information is received from memory via the 4-bit data bus.

The program counter can be loaded with an address contained in the direct address register section of the instruction register. This enables the processor to jump or branch from one part of memory to another to obtain its next instruction. The accumulator (A) is either loaded with a value from memory or a constant contained within the instruction. The accumu-



1. Internal structure of SAM – simplified architecture microprocessor.

lator can send data from the ALU, and its contents can be stored in memory.

The indirect address register (IAR) can be used to quickly indicate the memory location of data for a given instruction. The register can be initialised to some value by an instruction, and later used to locate data whenever it is needed.

There is also an indirect address buffer (IAB) – this can be used to save data addresses that are not currently being used for later transfer to the IAR. A single status flip-flop(s) is used to summarise the results of arithmetic and logic operations.

Instruction set

SAM is limited to the 16 instructions shown in *table 1*. This is because the 4-bit operation code used by SAM can only distinguish 16 (2^4) different instruction codes. Most 'real' microprocessors either have an 8-bit or 16-bit instruction code to provide a larger instruction set, but in this case, a 4-bit op code is adequate.

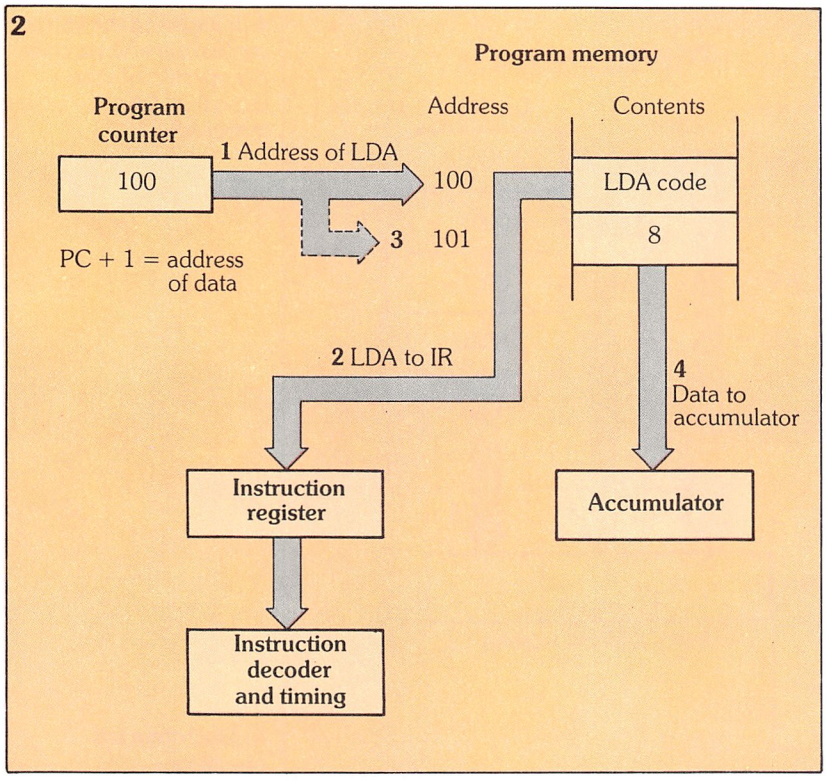
Data movement instructions

Data movement instructions are used to move data from the accumulator to

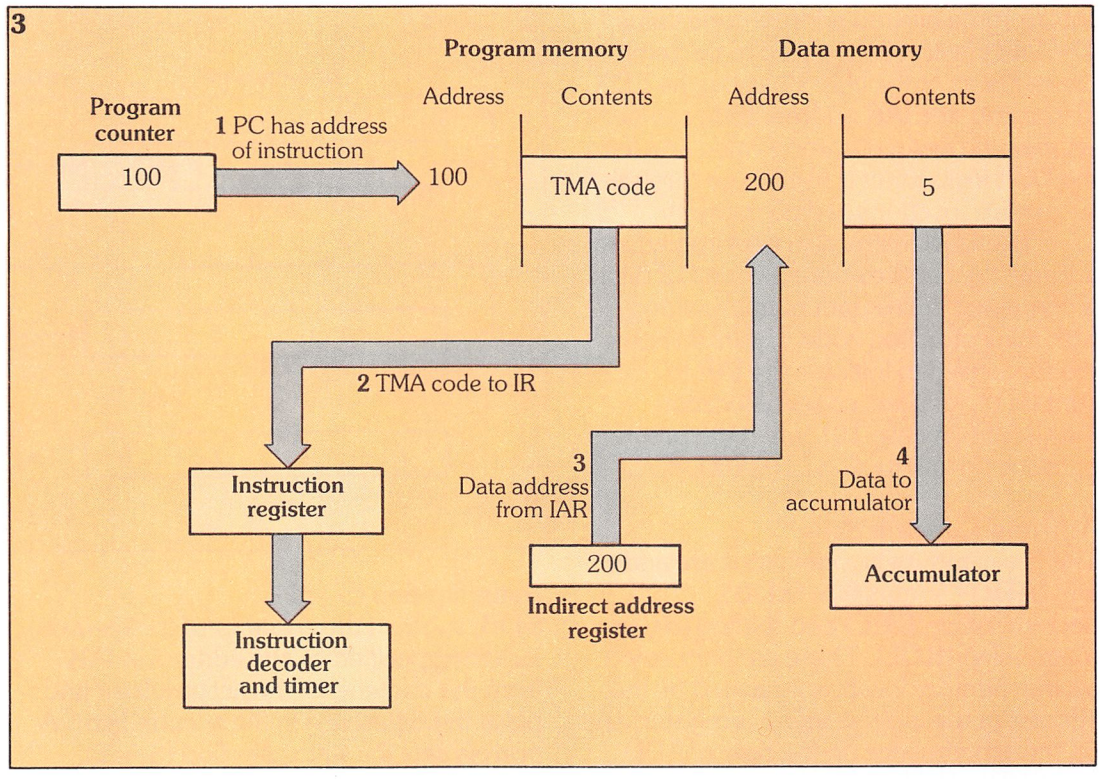
Table 1
SAM's instruction set

Instruction	Mnemonic	Operation	Affect on status
Arithmetic:			
Addition	ADD	A ← A + M	Status = Carry
Subtraction	SUB	A ← A – M	Status = Borrow
Decrement	DEC	M ← M – 1	Status = 1 if M = 0
Rotate left	ROL		Status = left bit rotated out of A
Logical:			
AND	AND	A ← A AND M	Status = 1 if A = O after operation
OR	OR	A ← A OR M	Status = 1 if A = O after operation
Data movement:			
Accumulator-to-memory	TAM	M ← A	None
Memory-to-accumulator	TMA	A ← M	None
Constant-to-accumulator	LDA n	A ← n	None
Input bit to status	IN	S ← Single data input value	Status = value of input signal
Output bit from status	OUT	Single data output ← S	None
Branching:			
Jump	JMP loc	PC ← loc	None
Conditional Branch	BS loc	PC ← loc if S = 1	None
Indirect address:			
Initialise IAR	LDX value	IAR ← Value	None
Decrement IAR	DEX	IAR ← IAR – 1	None
Exchange IAR and IAB contents	XCHG	IAR ↔ IAB	None

0≤n≤15: loc is an address value (12 bits); value is an address containing 12 bits; M indicates the memory data at the location indicated by the contents of the IAR



2. The LDA instruction uses immediate addressing in this sequence of steps.



3. Sequence of steps involved in using indirect addressing to specify the memory location of data.

to initialise the accumulator data to a given value (with the **LDA** instruction).

The LDA instruction uses immediate addressing, since the data is contained in the instruction. The sequence of steps is shown in *figure 2*. The other data transfers use the contents of the indirect address register to specify the memory location of the data involved (*figure 3*). The effect of these instructions can be summarised:

TAM: the contents of A are sent to the memory location specified by the address code in the IAR register.

TMA: the contents of the memory location specified by the contents of the IAR register are sent to A.

LDA n: the value n (0 to 15) is sent to the A register in binary form (0000 to 1111).

IN: the bit selected by the contents of the IAR register is sent to the status flip-flop.

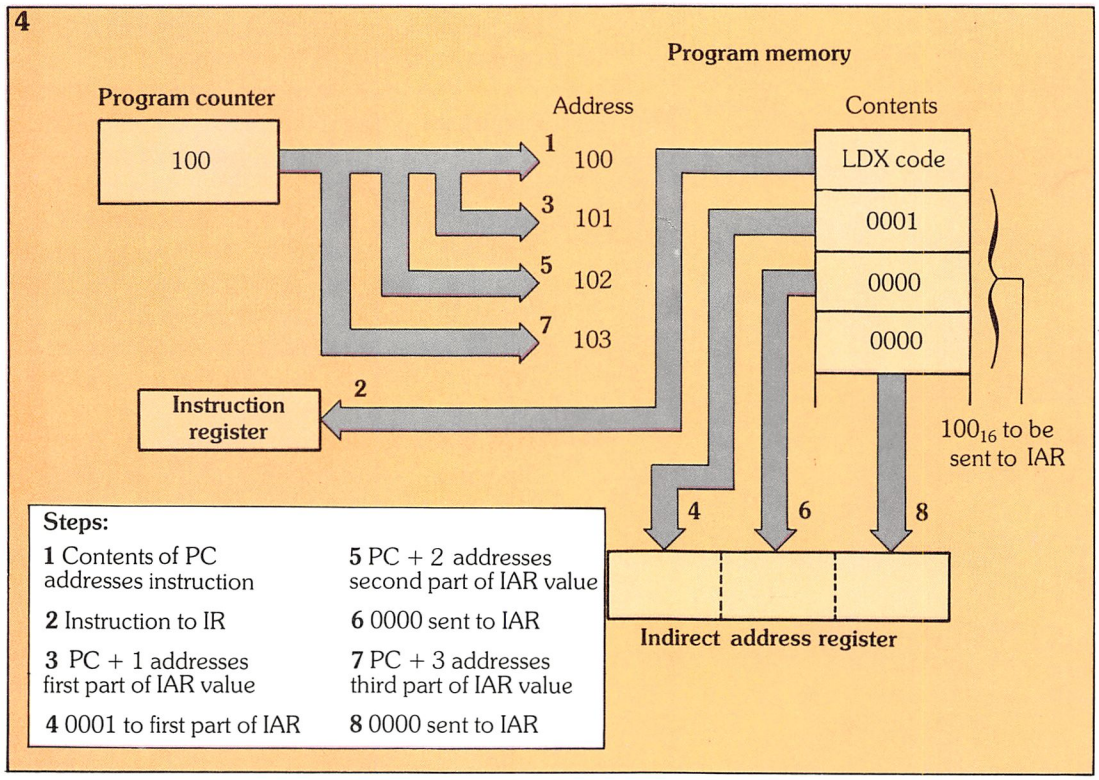
OUT: the contents of the status flip-flop are sent out to the external flip-flop selected by the address code in the indirect address register.

memory (**TAM**) or vice versa (**TMA**).

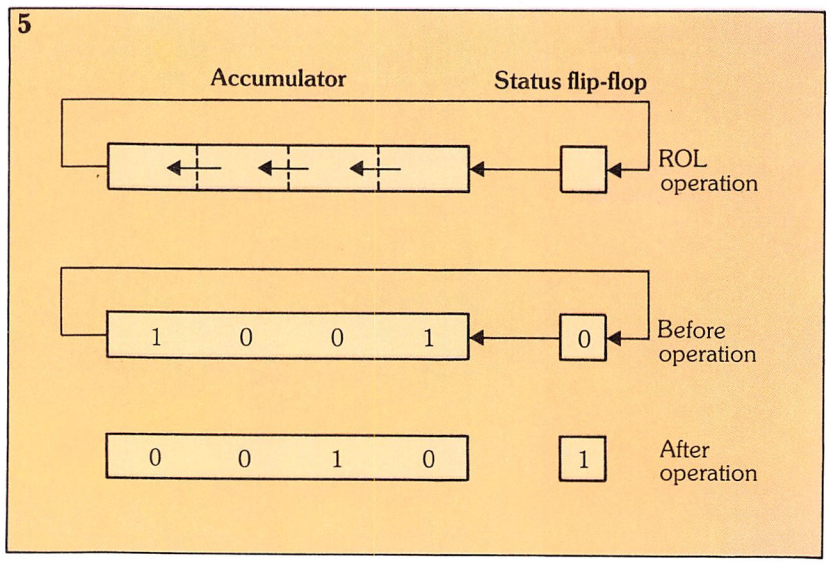
In the latter case, this might be to input or output a binary signal (to or from the status flip-flop) on the serial input and output lines (named **IN** and **OUT**), or

Indirect address register control instructions

Two instructions can be used to control the indirect address register. The **LDX** instruction is used to initialise the contents of the



4. Initialising the IAR to contain the address code of a specific location using the LDX instruction.



5. The rotate left operation.

IAR to some desired address code value. The **DEX** instruction is used to decrement the contents of the IAR. By using the LDX instruction, the IAR address code can be set to some desired starting value and then the DEX instruction is used to go through successive locations in data memory.

The LDX instruction uses immediate addressing, i.e. the address code (12 bits) is contained in program memory along with the instruction code. *Figure 4* shows the steps used to initialise the IAR to contain the address code for location 100_{16} (256 in base 10), when the program contains the instruction LDX 100.

Arithmetic instructions

The basic arithmetic operations are addition (**ADD**), subtraction (**SUB**), decrementing (**DEC**) and shifting left, or rotation left (**ROL**). These are performed on the memory location indicated by the address in the indirect address register, and on the accumulator (in the case of ADD and SUB). ADD, SUB, DEC and ROL operations are defined as follows:

ADD: add the contents of the memory location specified by the address in IAR to the contents of the accumulator. Place the sum in the accumulator. A carry sets the status flip-flop (S = 1).

SUB: subtract the contents of the memory location specified by the address in IAR from the contents of the accumulator and place the difference in the accumulator. A borrow sets the status flip-flop (S = 1).

DEC: decrement the contents of the memory location specified by the address in IAR. A zero result sets the status flip-flop. The results of the operation are stored in the same memory location.

The rotate left instruction, ROL, is

best explained with a diagram (*figure 5*). The contents of the accumulator are shifted one position to the left, and then the vacated right-most bit is filled with the value of the status flip-flop. Then the status flip-flop is set to the value of the bit shifted out of the left of the A register.

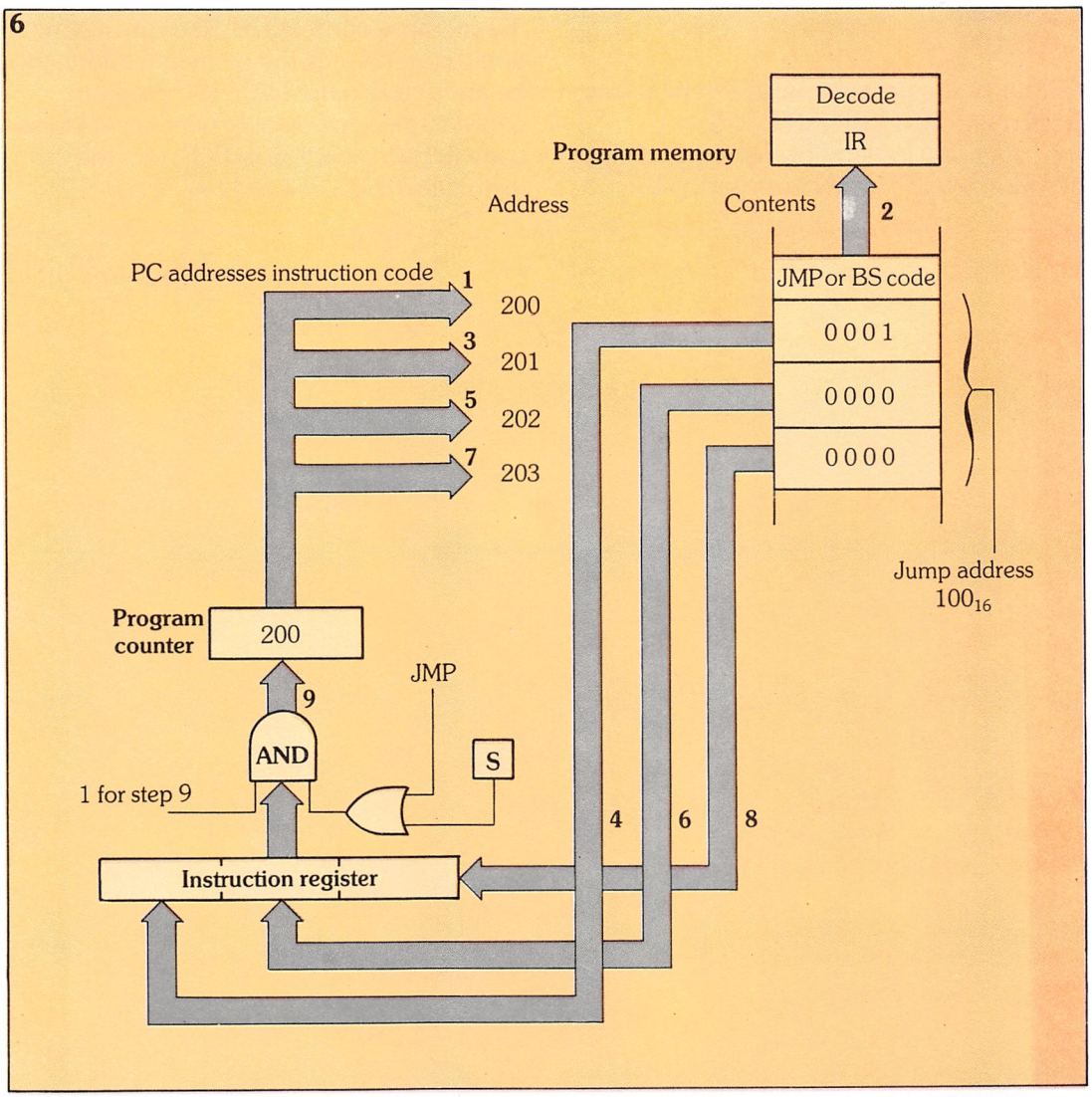
So, if the status flip-flop contains a 0 before the ROL operation, and the A register contains 1001 before the operation, the ROL will change the contents of the A register to 0010 and the contents of the status flip-flop to 1.

Logical instructions

The basic logical functions of **OR** and **AND** are provided by the instruction set. These operations are performed bit-by-bit on the contents of the accumulator and the memory location specified by the IAR register. The result is then sent to the accumulator, The status flip-flop is set to 1 if the result is a 0, i.e. if all bits of all results are 0s.

The AND operation is usually used for selectively clearing a bit to 0; the OR operation usually selectively sets a bit to 1.

6. Operation of a branch instruction.



Jump and branch instructions

The **JMP** instruction makes an unconditional jump to an instruction in another part of memory. This command takes the form: JMP location.

The location specified is the address to jump to – that is, the address of the instruction to be executed. This value is loaded into the program counter, and will cause the jump when it is executed. For

example, to cause the next instruction to be taken from memory location 100, the instruction will be:

JMP 100

A conditional branch instruction (**BS**) is also available. This will make a jump only if the status flip-flop contains a 1. So the instruction:

BS 100

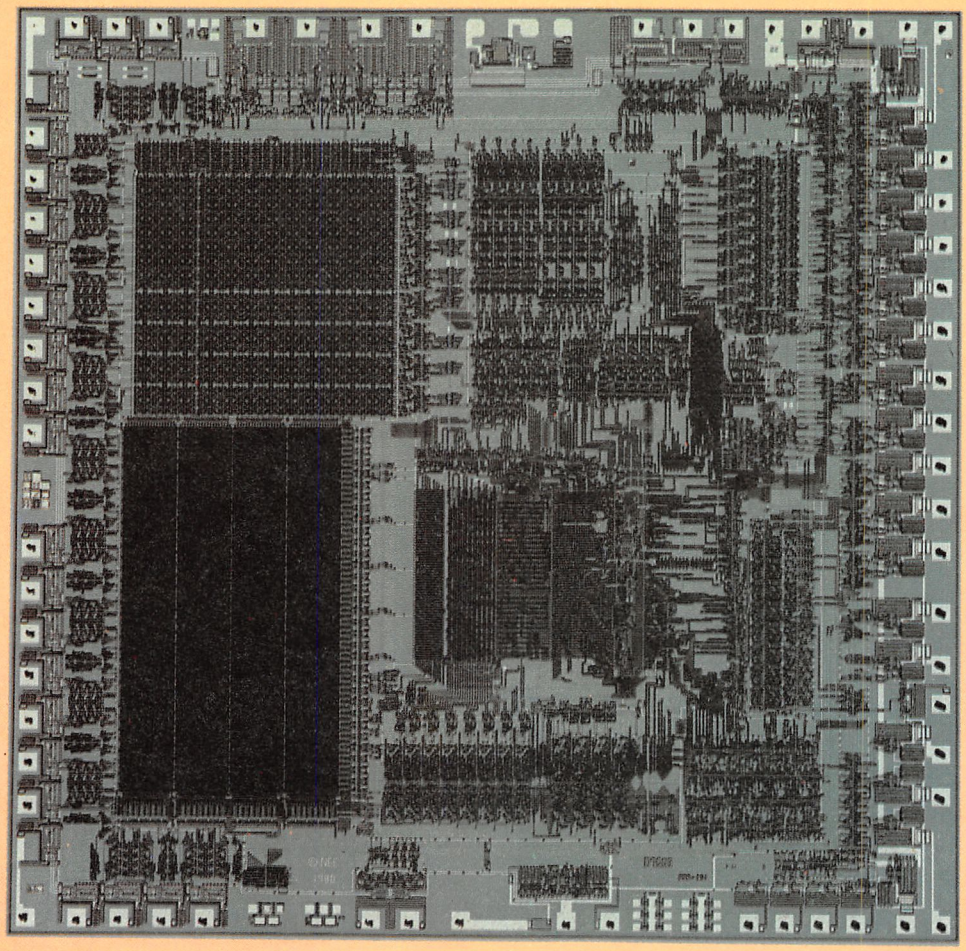
will cause the next instruction to be taken from memory address 100, if S = 1.

Figure 6 looks at branch instructions in more detail. The address part of the branch instruction is contained in the memory locations to the right of the instruction operation code. The microprocessor reads the instruction code and finds that the instruction is a branch (either JMP or BS).

Then the next three memory locations are read and the information sent to the memory address (DAR) of the instruction register. Once the complete address to be jumped is in the DAR, the program counter is loaded directly with this address code if the instruction is JMP; or if the instruction is BS, loaded only if S = 1.

7. Timing diagram for a memory read operation.

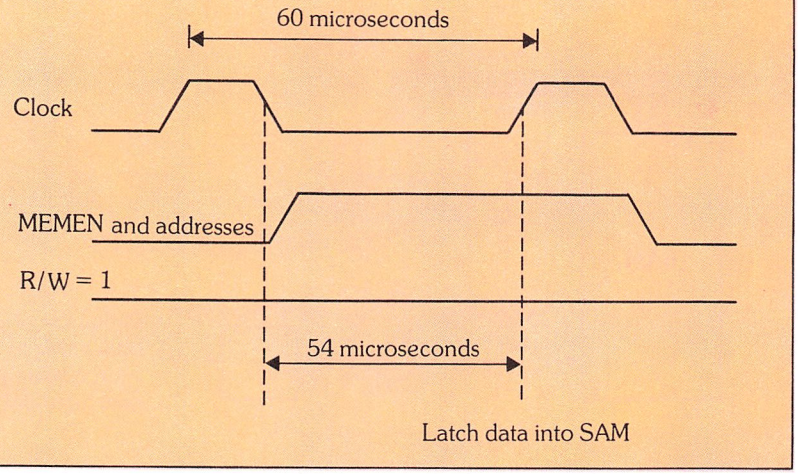
8. Memory write operation timing.



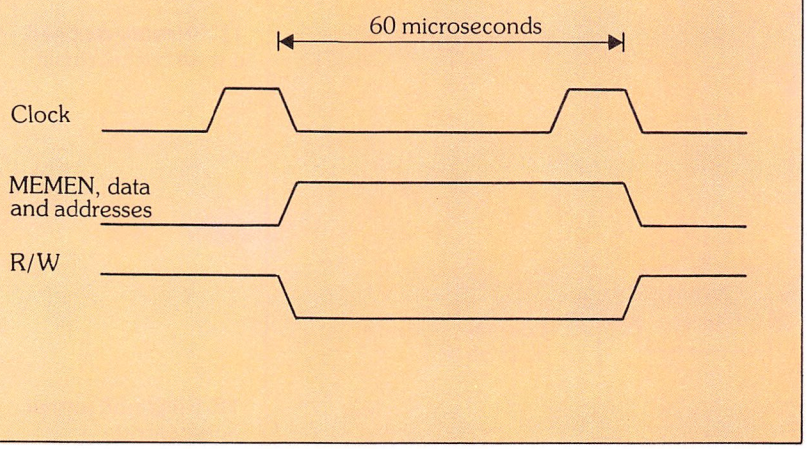
9. A memory mapped input device.

Left: photomicrograph of a 4-bit microprocessor.

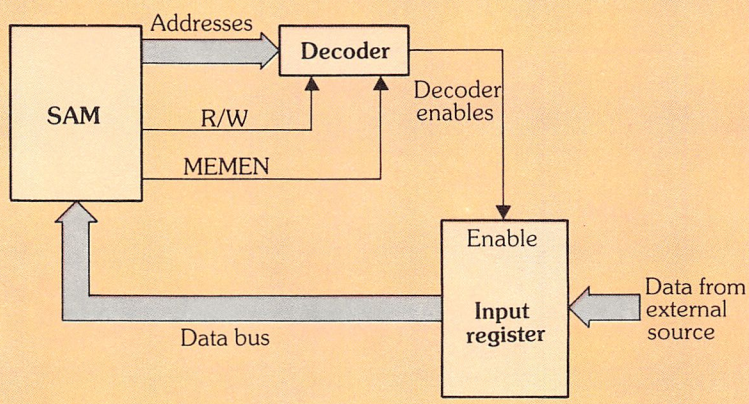
7



8



9



Timing features

We now need to look at the way SAM communicates with the external world, the microprocessor's communication operations are, of course, controlled by timing signals.

Memory read operation

A timing diagram for a memory READ operation is shown in *figure 7*. When memory is to be read, SAM outputs a memory enable signal (MEMEN = 1) to turn on the memory circuits. A signal is also sent out on the read/write line (R/W = 1) to indicate that a read operation is being performed. The addresses are sent to memory at the same time that MEMEN is set to 1, to ensure correct timing.

SAM inputs the data from memory on the leading edge of the clock pulse. So, the memory must respond with the correct data within the time between the trailing edge of one clock pulse and the leading edge of the next. In *figure 7*, this is approximately 54 microseconds which is more than enough time for semiconductor memory devices.

Memory write operation

When data is to be stored in memory, the address, data and memory enable signals are sent to memory at the same time (*figure 8*). The read/write signal goes low (R/W = 0) to indicate a write operation. In the case of a write, the memory must be able to store the information within the MEMEN time of 60 microseconds.

Input operations

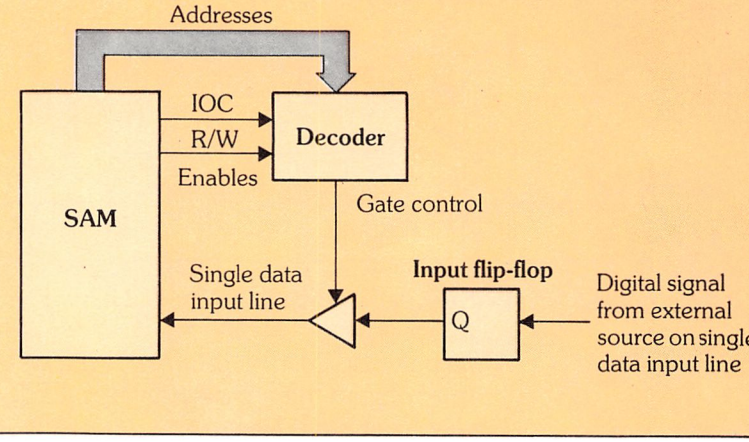
Figure 9 shows a data input device that is treated as if it is a memory location. This is known as **memory mapped input**. A 4-bit register is assigned a memory address and an address decoder is designed to generate an enable control signal in response to the correct address when R/W = 1 and MEMEN = 1. The input device places its data on the data bus when SAM wants the data read from the memory address given for that input device.

SAM can also support an **input port** by means of its IN instruction, as shown in *figure 10*. When this instruction is executed the address in the IAR is sent out on the

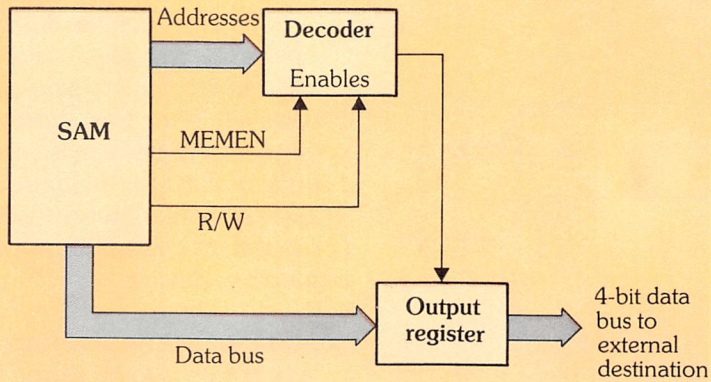
address lines, the input/output control (IOC) signal is generated and the read/write signal is held at 1.

In order to provide the single bit that this instruction expects, the address is decoded, and the input signal from a selected flip-flop is gated onto the microprocesor's single-bit data line. This single bit is sent to the status flip-flop (S) inside SAM. After the IN instruction, S contains the same binary value that the selected input flip-flop has stored. This special purpose input scheme is useful in testing single-bit digital signals throughout the system, and in setting up serial data transfers from other systems.

10



11



10. Single-bit input port control circuit.

11. Memory mapped output control circuit.

12. Single-bit output port control.

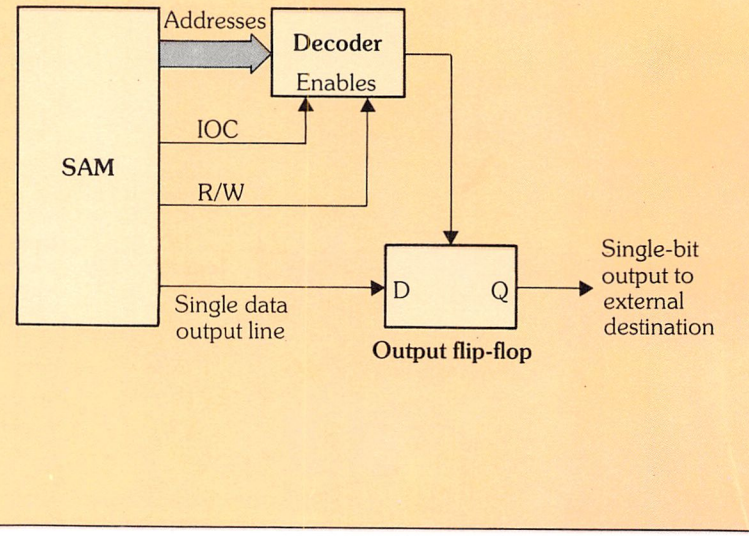
Output operations

SAM also supports memory mapped outputs and single-bit outputs (*figures 11* and *12*). As with the input operation, a 4-bit register is used as a memory location. However, it now *receives* signals from SAM and stores outputs.

A decoder that detects the address and the memory write conditions (MEMEN = 1, R/W = 0) outputs a control signal that latches the data on the data bus from SAM into the output register. For single-bit outputs, the output control signals (IOC and R/W) are used with the addresses to provide a control signal that latches the data into the output flip-flop. The data comes out from the status flip-flop onto the microprocessor's single-bit data output.

The system's timing features enable memory and I/O blocks to be connected to SAM to form a smoothly operating system.

12



Building a system

SAM's hardware features and instruction set are complete enough to satisfy the needs of many applications. We'll now solve an example problem to illustrate the power of this simple microprocessor.

We shall assume that SAM is to be used to receive a total of 32 binary data bits at one time (serially)on the single-bit input line. This 32-bit binary number must then be converted to its decimal equivalent, which consists of 10 binary coded decimal (BCD) 4-bit groups. These numbers are then to be output on LED seven-segment displays.

must be corrected.

Figure 13 shows one way in which an illegal code could be generated – a two decimal digit number is shifted one position to the left. This should effectively multiply the number by two! If the BCD number is 0000 0100 (04) as shown in *figure 13a*, then a left shift will multiply it by two – giving 0000 1000 (08). This contains only legal BCD codes and is correct.

However, in *figure 13b* a BCD number of 0000 0101 (05) is shifted left – giving 0000 1010 which contains the illegal code, 1010. The correct BCD code is 0001 0000 (10). This would have been obtained if 0011 had been added to the 0101,

13

		Before left shift	After left shift	
a) $2 \times 4 = 8$		0000 0100	0000 1000	Correct
b) $2 \times 5 = 10$		0000 0101	0000 1010	Incorrect (0A)
	+	0000 0011		
		0000 1000	0001 0000	Correct (10)
c) $2 \times 8 = 16$		0000 1000	0001 0000	Incorrect ($10 \neq 16$)
	+	0000 0011		
		0000 1011	0001 0110	Correct (16)

13. Generating an illegal code when a left shift operation is performed on a two decimal digit number.

Table 2
BCD to decimal conversion

Decimal Digit	BCD Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1010

BCD numbers

In order to understand this problem fully, we'll revise what we know about BCD numbers. As we should know, BCD code is simply the 4-bit binary equivalent of the numbers 0 to 9 (*table 2*). Any code other than those listed is **illegal** (such as the binary equivalent of 15 which is 1111) and

before the left shift. Then the resultant code 0000 1000, shifted left would give 0001 0000 (10).

In the same way, the BCD code 0000 1000 (08) shifted left would give us 0001 0000 (10). This is legal BCD code, but the wrong answer. Again, the correct answer would have been obtained if 0011 had been added to the 0000 1000 before the shift left – giving 0000 1011. Shifting this left gives us 0001 0110 (16),which is correct.

So, we can see that if the BCD code for a digit is 0100 (4) or less, then the resulting code will be correct after a left shift. However, if the code is 0101 (5) or greater, 0011 must be added to that code prior to shifting. This rule forms an algorithm that SAM can be programmed to follow whenever a BCD number is to be shifted one position left.

Binary to BCD conversion

Figure 14 shows the procedure for binary-to-BCD conversion in a simple flowchart.

The binary number to be converted in this case is of 32 bits, so the counter is set to this value. The largest binary number that can be represented in 32 bits of memory corresponds to a ten digit decimal number. Therefore, 40 bits of storage are needed to hold this number in BCD code, as each BCD digit is a 4-bit group. *Figure 15a* shows the BCD bits along with the 32 bits used to hold the binary number, arranged as a 72-bit 'number' in the system memory.

SAM is a 4-bit microprocessor, so it handles the 72-bit group as 18, 4-bit numbers – comprising 10 BCD digit groups and 8 binary number groups. The procedure used to convert the 32 bits of binary data into BCD is as follows:

1) input the binary number and store in the 32-bit binary number locations;
2) clear the BCD digit groups to 0000;
3) set the binary bit counter to 32;
4) repeat the following sequence of operations 32 times:
   a) add 0011 to any BCD code greater than 0100 (BCD coded 4-bit groups only);
   b) shift the 72-bit combined BCD-binary number to the left one bit position;
   c) decrement the counter.

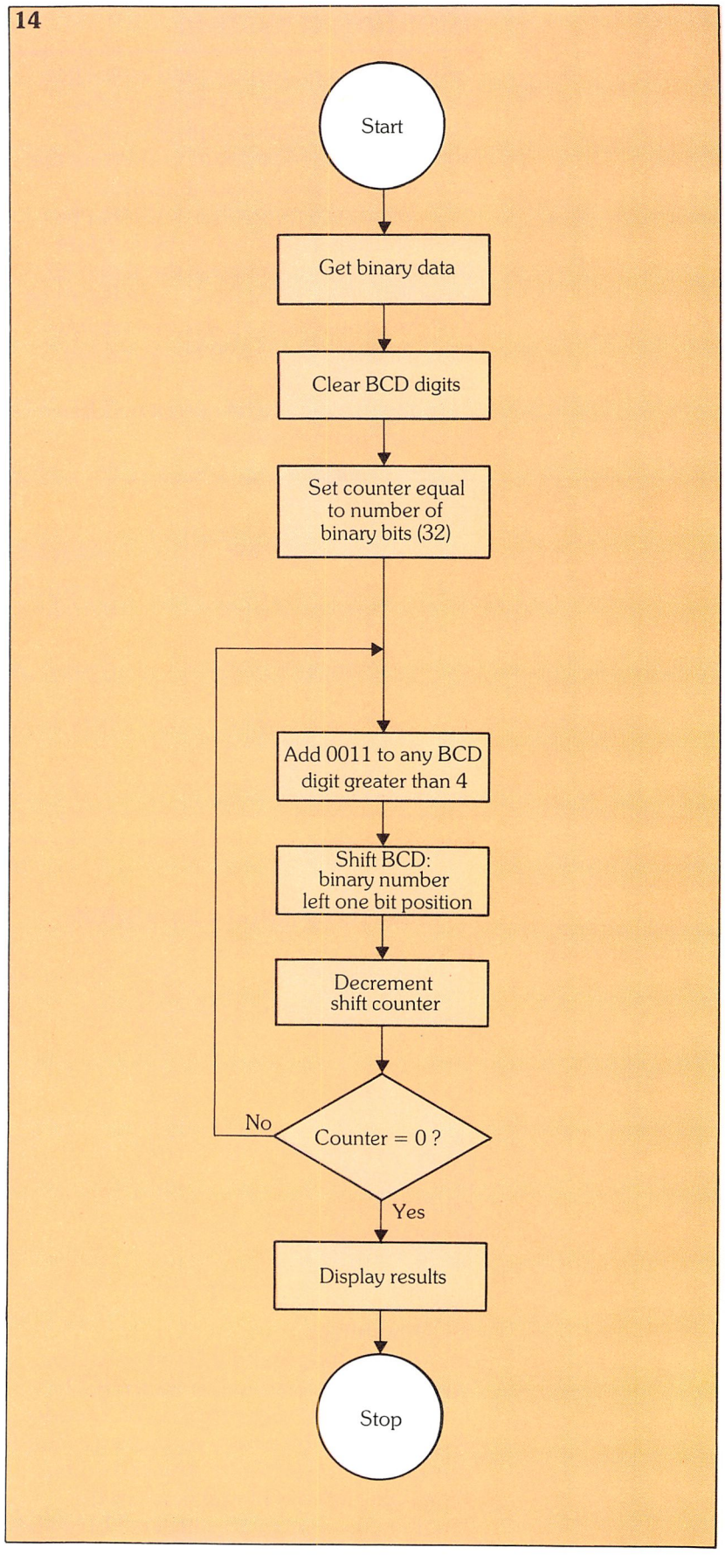
After the 32nd shift, the BCD equivalent of the original binary number is contained in the 10 BCD digit storage locations. From there, the BCD numbers can be output as decimal numbers, using seven-segment displays linked to decoder/drivers. *Figure 15b* shows an example of the conversion process, using only 4 bits.

Basic serial data transmission

One of the tasks the microprocessor has to perform in our example problem is reading in and storing the 32-bit binary number. This is performed serially, and one common serial data format is shown in *figure 16*.

Eight bits are sent at a time, beginning with a start bit and ending with one or two stop bits. The system that receives this signal must monitor the signal line to determine when the start bit occurs. It must then determine the centre of the start bit interval, and determine the value of the eight data bits by checking the input line at appropriate intervals. Finally, a stop bit

14

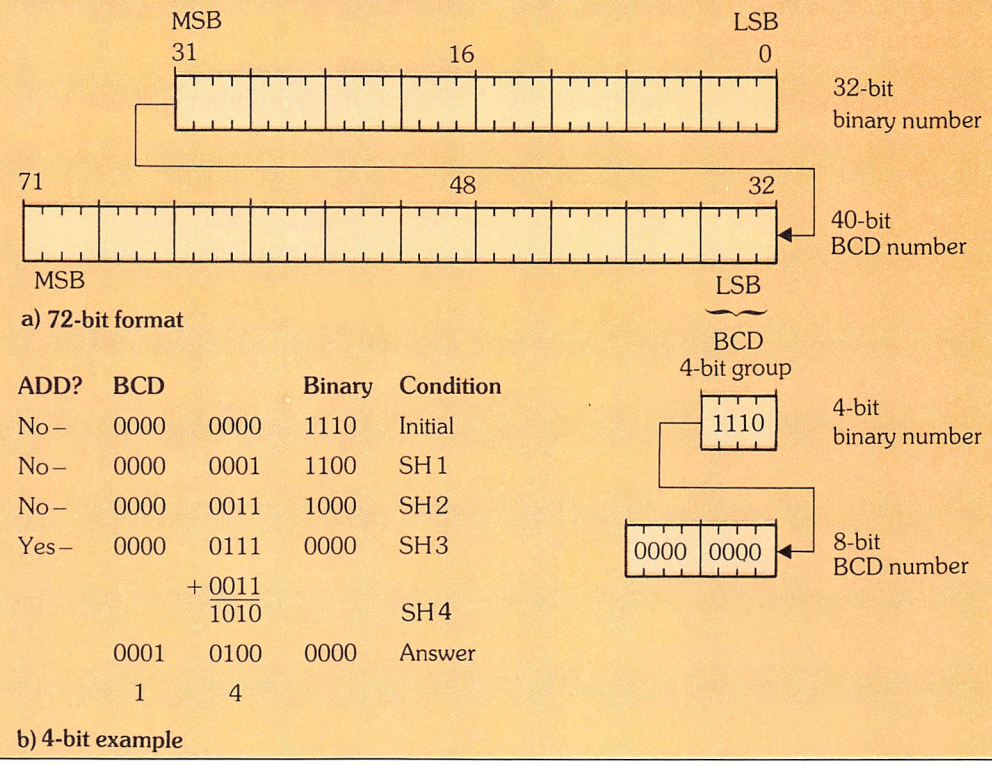


14. Binary-to-BCD conversion flowchart.

15. 4-bit binary-to-BCD conversion.

16. A serial data format.

15

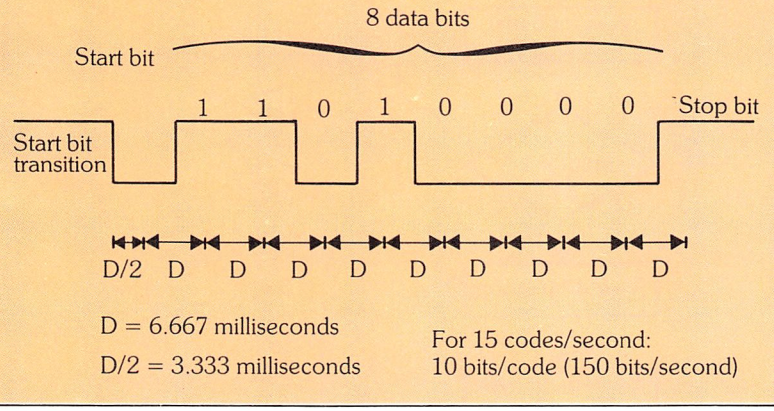


a) 72-bit format

ADD?	BCD		Binary	Condition
No–	0000	0000	1110	Initial
No–	0000	0001	1100	SH1
No–	0000	0011	1000	SH2
Yes–	0000	0111	0000	SH3
		+ 0011		
		1010		SH4
	0001	0100	0000	Answer
	1	4		

b) 4-bit example

16



must be detected, after which data transmission is known to be over for that 8-bit group.

The interval times involved in this type of reception depend on how many 8-bit groups (bytes) are sent per second. A total of 10 bits are sent for each 8-bit data byte. Therefore, if 15 such bytes are sent per second, the bits would be arriving at the rate of 150 bits s^{-1}. Each bit interval is 0.0067 seconds (6.67 milliseconds). Each 8-bit data group is received in this way and four 8-bit data groups like this are used to send the entire 32-bit binary number to SAM for conversion.

Summary of tasks

Taken individually, the three tasks that our system is to perform are relatively simple. The processor must input the serial data and send it to the microcomputer memory; determine when all 32 bits have been received, and then convert this binary number to its decimal equivalent; display the decimal numbers and wait for the next binary number to be received.

The total system is simply implementing the basic operations of **sense** (received the input binary number), **decide** (decide on its decimal equivalent), and **act** (output the result to the display devices). SAM, with the appropriate additional hardware devices, will perform these tasks according to instructions contained in a program.

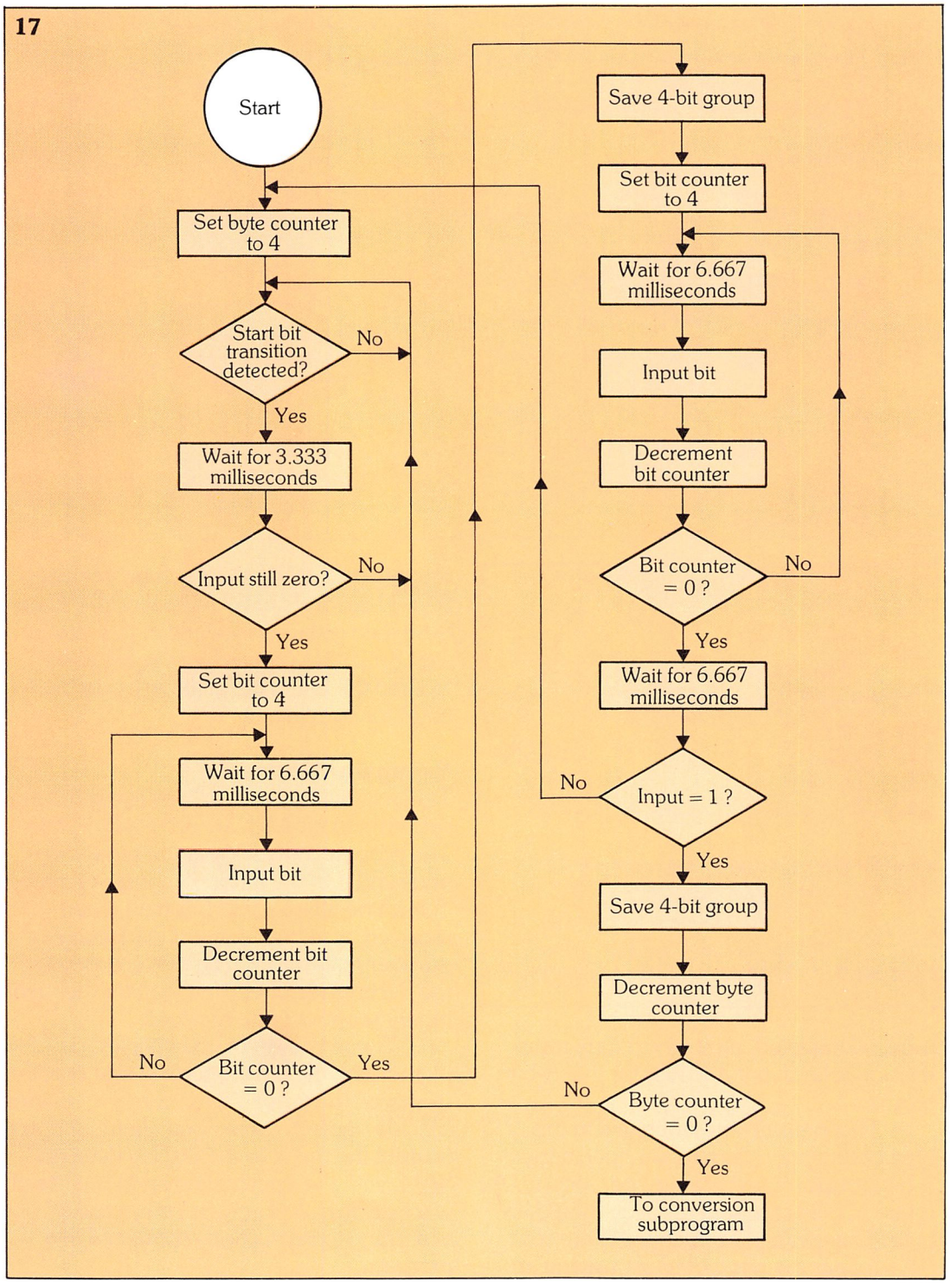
Developing the program

The first thing to do is break the problem down into subprograms, each of which handles a specific task. For example, the number conversion program breaks down into:

1) **Input subprogram** – receives the serial data and stores it in memory.
2) **Conversion subprogram** – converts the binary number into its decimal equivalent.
3) **Output subprogram** – outputs the decimal codes to the display devices.

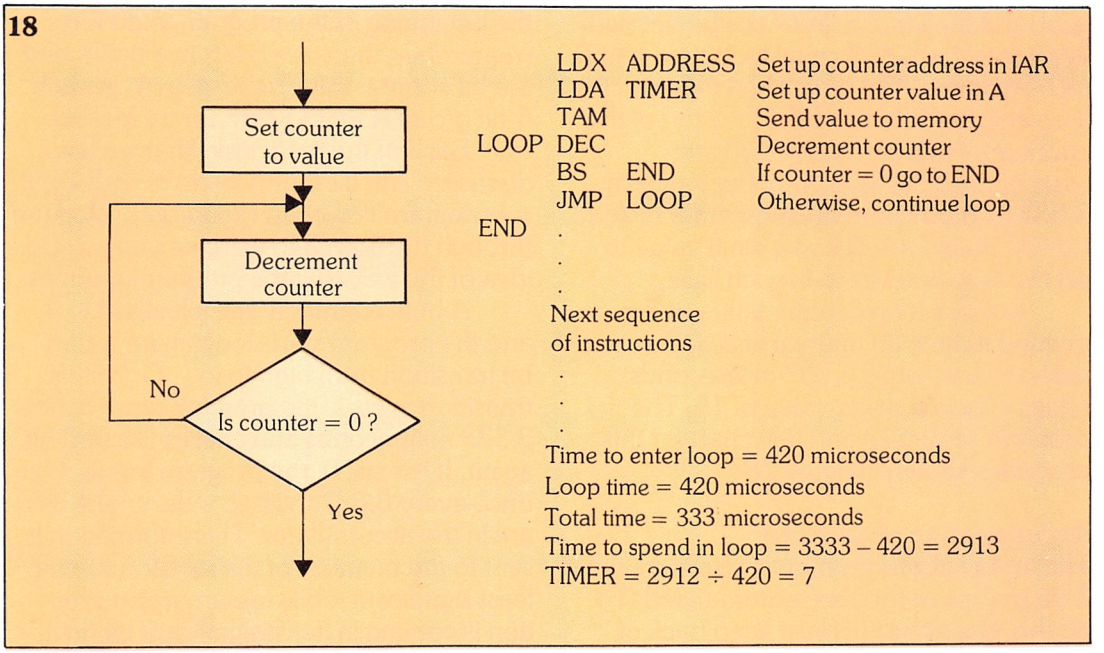
Input subprogram

The input subprogram is shown in *figure*

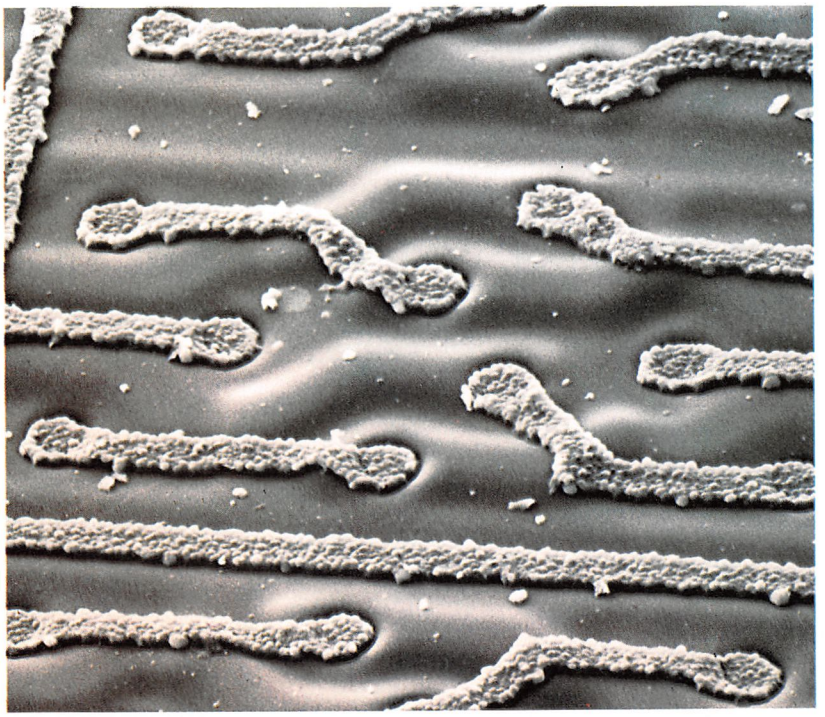


17. Input subprogram flowchart.

18. Flowchart and program sequence for the interval timer.



Below: scanning electron micrograph of part of the surface of an ITT 4116 16K dynamic RAM chip. ×370.



Science Photo Library/Dr. Tony Brain

17 in flowchart form. It must perform the following operations:

1) Determine when the start bit transition occurs, by continually checking the serial data line for a zero. Once a zero is detected, the processor can go to the next input operation.

2) Determine the mid-point of the start bit, checking for a zero, 3.333 milliseconds (for a 150 bits s^{-1} signal) after it has detected the start bit transition. If the input line still has a zero on it at this time, the subprogram can start looking for data bits. If not, the subprogram will have to return to the first task of checking for a start bit transition.

3) Once the start bit has been verified, the program inputs a data bit once every 6.667 milliseconds, until 8 data bits have been received and stored in memory.

4) After 8 bits have been received, the program must wait 6.667 milliseconds, to see if the input is one. If it is, the transmission terminates properly, and a byte has been received. If the input is zero, the transmission is incorrect and the program must request a retransmission of the same byte.

Steps 1 and 4 are repeated until all four bytes of binary data have been received. Then the conversion subprogram can be used.

Timer instruction sequences.

The input subprogram must provide the basic operation of a timer. First it has to provide 3.333 millisecond time intervals to find the centre of the start bit. Second, it must be able to time 6.667 ms in order to determine the interval between data bits.

A program sequence that provides a timing operation like this consists of a counter and a program loop that continues to decrement the counter until it is zero. By working out how long the system takes to

go round the loop, a timer can be devised. We know that SAM's total loop time is 420 microseconds. The time taken to set up the loop counter value and the location of the counter in memory is also 420 microseconds. If this value is deducted from 3.333 or 6.667, and the remainder divided by 420, then we will know what value to set the loop counter to in each case.

In fact a loop counter value of 7 is needed to time an interval of 3.333 milliseconds, while 6.667 milliseconds needs a loop counter value of 15. The program sequence and flowchart for this process is shown in *figure 18*.

Accumulation of data bits

The input subprogram must also store the 4-bit groups as they are accumulated. To do this, the program must keep track of how many bits have been received, so that when 4 bits are fed in they are stored in memory as a complete 4-bit group. Assuming that successive memory locations are used to store these groups, the program must also keep track of the next location to be filled.

The program also has to keep track of the total number of bits that are sent as input, so it knows when the 32 bits have been stored. A byte counter is set to 4 at the beginning of the program to do this (remember, there are 8 bits in a byte), and the bit counter must be reset to 4, as each 4-bit group is sent to its memory location.

Each of the basic tasks that we have discussed can be found in the overall input subprogram flowchart (*figure 17*). Working through the flowchart will give you a good idea of the way that the program operates.

A byte counter is first initialised to 4, and the program starts looking for a start bit transition from high to low. Once this transition occurs, the program timer counts 3.333 milliseconds and checks the start bit again. If it is still 0, the program inputs a bit once every 6.667 milliseconds, until 4 bits are in the accumulator. This information is sent to the memory of the location for the least significant 4 bits (assuming information is coming in, least significant bit first).

Again, the program inputs every 6.667 milliseconds until the next 4 bits are in the accumulator. This information is sent to the next 4-bit memory location. Since 8 bits have been received, the byte counter is decremented and the procedure is repeated until all 4 bytes have been received and stored in memory.

At this point the work of the input subprogram is complete and the conversion subprogram can begin operation.

Glossary

DAR	direct address register
IAB	indirect address buffer
IAR	indirect address register
IR	instruction register
PC	program counter
SAM	simplified architecture microprocessor
timing loop	program loop that is used to time the microprocessor's operations. Once the loop time and the time taken to initialise the counter are known, the loop can be set to any multiples of these

12

COMPUTERS & SOCIETY

Data protection

Why is it necessary?

Information is held about members of the public in all sorts of places for many different reasons; examples of the kinds of public institutions and private companies that maintain files on individuals are given in *figure 1*. Until recently, this information was held in manually operated filing systems. These were large and cumbersome and extracting information was a time consuming operation. Consequently, the transfer of data between one public or private sector database and another, and even the linking of information within one database, could not be handled efficiently, and in practice, it didn't often happen.

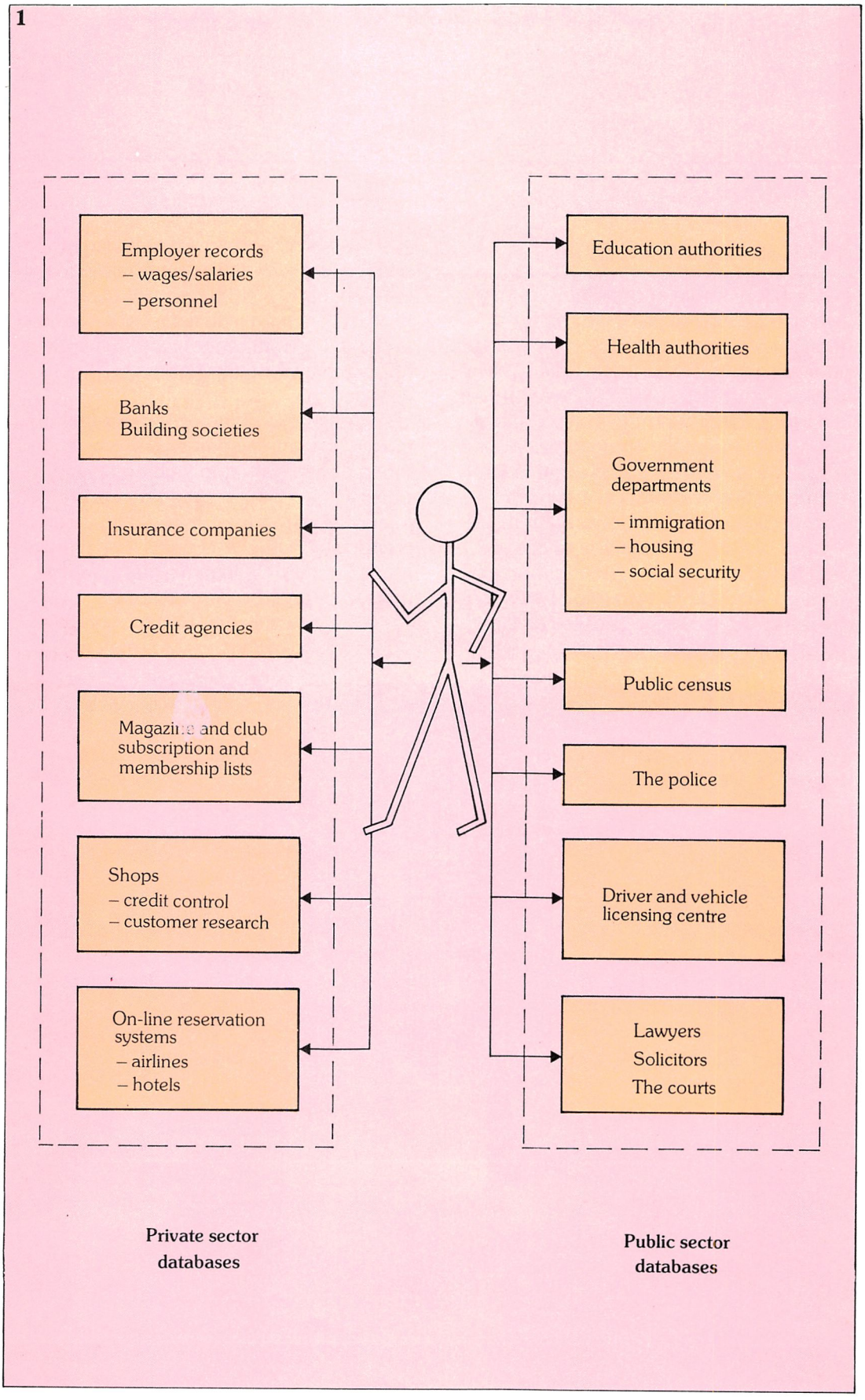
As a result, the confidentiality of information was more or less taken for granted. However, the advent of computerised data storage and retrieval systems and the development of extremely powerful search and linking algorithms has changed this.

Advanced communications techniques which enable databases to be linked, mean that information held on one computer for one purpose may be transmitted to another computer, so building up a composite 'personal profile'. The danger of this information being misinterpreted or misused is now causing concern.

For example, in 1980 a Canadian-born Indian was arrested for making an

Science Photo Library/Jerry Mason

Right: personal identity numbers provide one mechanism for limiting access to certain information. Here, various security codes are displayed on the screen.



1. Information is held about us in many different places.

illegal left-hand turn while driving. A check on police files held across the U.S. (carried out in seconds by computer) discovered that he had gone absent without leave from the Marine Corps ten years previously. He was held for five months before it was discovered that he had left the Marine Corps under a special discharge scheme!

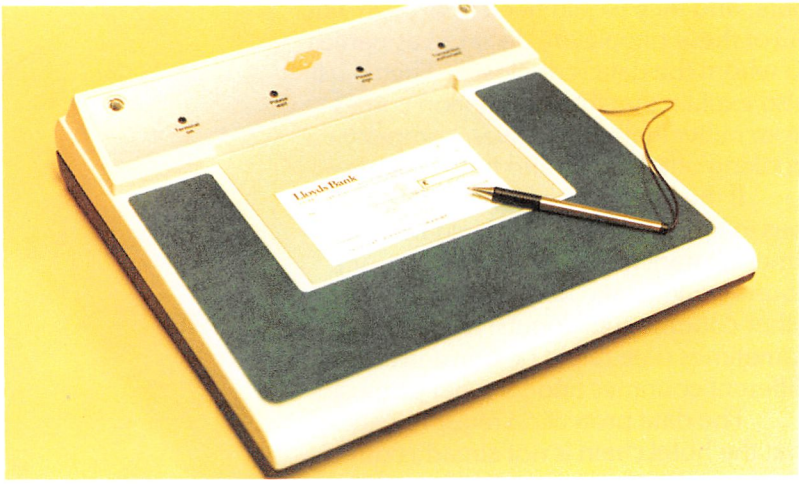
This example illustrates another problem – that the electronic storage and handling of personal information is now being entrusted to automatic equipment. Of course, errors also occur in manual systems, but automatically linking computer databases just exacerbates the problem.

There may be mistakes in programs; operators could key in incorrect information; the information may not be frequently updated; or there may be insufficient data to give a clear picture – often, what is not held is just as important as what is held.

Of course, private individual concern will never be the prime motivation for formulating government policy on such matters. The growing need for confidentiality to be maintained both for industrial and government secrets – where money or perhaps national security is at stake – has forced successive governments to look at the problem although, until recently, nothing concrete had been achieved.

So we can see, then, that there are two aspects to this problem of data protection: **privacy**, which is concerned with avoiding the misuse of information; and **security** – the means whereby privacy can be ensured. We will deal with each in turn.

Below: this signature verification terminal provides another form of security measure. The signature is matched by computer against one held in memory.



The Research House/Transaction Security

Data privacy

It is now generally accepted that with the ever increasing *amount* of information that is being held, and the lack of public awareness regarding the nature and *accuracy* of that information (together with a lack of knowledge as to *why* it is being stored), some kind of public accountability and 'right to know' should be incorporated into the law.

After a decade of discussion, several white papers and two commissions (the Younger Committee in 1972 and the Lindop Committee in 1978), it was the signing of the Council of Europe Convention on data protection in May 1981 that forced the government's hand into action.

The Council of Europe Convention sought to achieve, for all individuals in all member countries, respect for their freedoms, especially their right to privacy, in regard to the automatic processing of their personal details. The Council laid down minimum legislation for ratification of the treaty. With this in mind, the Data Protection Bill was introduced in 1982, modified in 1983, and is now coming into effect in 1984.

There are very strong economic reasons for ratifying the Convention. It is likely that those countries which have enacted legislation might restrict the flow of information to only those countries which have similarly passed some form of data protection law. Otherwise, they argue, 'data havens' might spring up, where personal information could be processed without public scrutiny. As Sweden, West Germany, France, Norway, Denmark, Austria, Belgium and Luxembourg all possess legislation, it was imperative for the U.K. to enact laws quickly.

The Data Protection Act

To understand the Data Protection Act, several terms must first be defined: **data users** are people, or organisations who are in control of the contents and use of automatically processed personal data; **personal data** is information about living people, who can be identified by that information, which is held on computer. It includes expressions of opinion about people; **data subjects** are those people about

Science Photo Library/Hank Morgan

Left: computers with high speed parallel processors, such as this one at the Exxon research laboratory, are used for such things as oil exploration. The kind of information obtained would be of great interest to competitors and must be protected.

whom there is information held on a computer. These are the people who are to be protected under the new Act; **computer bureaux** are organisations which exist to process data, including personal data, for other organisations or people.

The Act sets up a **data protection register**, run by a government appointed registrar. All holders of personal data must now register their use of such data, and adhere to eight basic principles of behaviour. If they ask, all data subjects have the right to know that personal data is held about them, and what it is. If it is inaccurate, it must be corrected or removed. They also have the right to claim damages if they are harmed as a result of data being inadequately protected or wrong. There are, of course, quite a few exceptions to this right of access, which will be dealt with later.

The data user registers by providing the registrar with the company name and address, and a description of the data and the purpose for which it is held. The data user must also explain where the data came from, who is allowed to see it and which countries, if any, it is being transferred to. They must also supply an address where data subjects can apply for access.

So, for example, a hotel chain might keep a list of forward bookings on computer. They would have to apply to the registrar, giving name and address, and explaining that names, addresses, telephone numbers and sex of future guests are kept in order to arrange rooms. They would need to explain that management and bookings and accounts clerks see this data, and they would need to explain why. All of these people are declared as parties to whom the data is disclosed. They must also explain that the information comes from individuals themselves, company booking office clerks and travel agents.

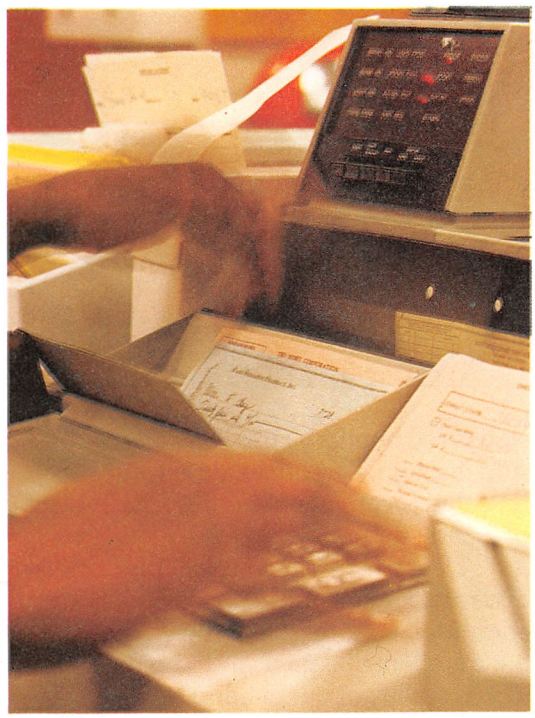
If the head office of the hotel chain was in Paris, personal data might sometimes be transferred there; hotels in one country might take bookings for hotels in other countries etc. It must be declared that sometimes data might be transferred to different countries and for what purpose.

The data protection registrar checks that the hotel chain is not already con-

travening the Act. Once registered as a data user, the following eight principles must be complied with.

The 8 principles

1. Data must be obtained 'fairly', i.e. the data user must only obtain information from either those who are allowed to have that information in the first place, or those who have been required to supply it by law or international obligation. Data must also be processed fairly. It is also important that data users do not mislead data suppliers as to the purposes of data collection.
2. The data which is collected must only be held for the one or more lawful purposes for which the data user has registered. If a travel agent, for example, has registered that it collects certain personal data in order to make holiday bookings, and suddenly decides it could also collect data for market research, it would be breaking the second principle if it were to do so. Every purpose must be registered.
3. Personal data must only be used for the purpose for which it has been registered. So our travel agent could not use the data it is collecting for making bookings, which is a registered purpose, for market research, which is not. It would have to register market research as another purpose, and pay the appropriate fee.
4. The personal data which is held should be adequate, relevant and not excessive, for the purpose for which it is collected. The terms 'adequate', 'relevant' and 'not excessive' are, of course, open to various interpretations. But if a travel agent were to collect information regarding the salaries of its clients, this would be deemed irrelevant and excessive.
5. Personal data should be accurate and, where necessary, kept up-to-date. This is particularly important. If individuals are harmed due to inaccurate or out-of-date data, claims for damages can be made against the data user.
6. Personal data should not be kept for longer than necessary.
7. Personal data must be made available, on request, to data subjects, but only the data relating to that individual. Data users may charge a fee for this. Also the information must be in a form which the data subject can understand.
8. Data must be protected by the data user against unauthorised access, alteration, and accidental or deliberate disclosure, destruction or loss.



Science Photo Library/Allen Green

Right: automatic cheque processing equipment in use at a New York city bank.

Any complaints against data users are made directly to the data protection registrar who investigates and then enforces the Act through various measures. The registrar can serve enforcement notices on any data users breaching the principles: if data is inaccurate, the registrar can demand that the data be erased, in accordance with principle 5; if the data has been unfairly collected (principle 1), or held for too long (principle 6), the registrar can again demand that the data be erased.

The registrar also has the power to prevent data users transferring data abroad to countries either not adhering to the Council of Europe Convention or where data is being processed unfairly.

As a final sanction, the registrar may serve a deregistration notice. This means that the data user is no longer lawfully able to process data automatically. Data users have the right of appeal to the Data Protection Tribunal (comprising lawyers, computing specialists, computer users and laymen, appointed by the Secretary of State and the Lord Chancellor); there is a further right of appeal to the High Court.

Science Photo Library/David Parker

Left: in a large organisation there will need to be many different levels of authorised access to the information held in the computer database.

Exemptions to the Act

There are certain areas where this law does not apply at all. Total exemption is given to all records held in the cause of national security, for example. After much pressure from the business community, personal data held *only* for payroll and accounting purposes also does not need to be registered.

Personal data collected and processed on a word processor is also exempt. This makes sense because the data being used to produce letters and documents is always changing, and none of it is likely to be kept for very long, simply because of the cost of storing it all. In fact, any personal data that is only to be used for text production is exempt from registration. Mailing lists only containing names and addresses are exempt from the data protection law. This will have annoying, if harmless, consequences. The piles of 'junk mail' falling through letterboxes from unknown sources is frequently the result of mailing lists being bought and sold.

Clubs do not have to declare membership data which is held on computer, provided that the club is small (unincorporated) and its members agree to their names being placed on the computer. The club is still bound by the rules on disclosure, though: personal data cannot be revealed to anyone other than the subject of that data.

Data which is held solely for domestic and recreational purposes, as on a home computer, is exempt. If this were not so, the numbers of data users to be registered would really get out of hand.

Finally, information relating to judicial appointments, groups offering financial services to clients, research and statistics where an individual could not be identified, and back-up data is exempt.

One of the most controversial sections of the Act states that data need not be revealed if it is held for the purposes of preventing or detecting crime, catching or prosecuting offenders, assessing or collecting taxes or duties or controlling immigration. It will be very difficult to define the dividing line between that information held by the inland revenue or police and immigration authorities for the purposes just stated and data which would not prejudice such investigations, that could be disclosed. The Home Office has denied that the police, for example, will be exempt from the provisions of the Act and much of the information held on the Police National

Computer System will be accessible to the registrar and data subjects.

Another thorny issue yet to be resolved concerns medical files and it is likely that the Home Office will further exempt data held by the local authorities and voluntary organisations etc. that maintain information regarding the physical or mental health of a data subject.

A further problem regards the right of access to information by persons other than the data subject, the police or inland revenue, for instance, and whether the data subject should have knowledge of that access. Balancing the need to guarantee individual privacy with the needs of the rest of society is a difficult thing to do.

Below: with so many people now having access to computer databases, the prevention of computer fraud is a real concern.

ZEFA

Weaknesses of the Act

With such a provocative subject, there is bound to be disagreement as regards the effectiveness of the new legislation. There have been claims that the Act does not go far enough, and that it might not satisfy the Council of Europe Convention. On the other hand, it has been claimed that individual freedoms have been placed at risk and that the new law offers no protection for individual privacy at all.

In fact, the Act does have weaknesses – not least because it totally ignores the proposals of the Lindop Committee which was specifically commissioned to propose new measures for data protection. The Committee suggested the setting up of Data Protection Authority to oversee the development of individual codes of practice for each profession and industry. Instead, the government accepted the eight principles postulated by the Younger Committee. These only deal with the private sector – the public sector cannot be sued for failure to comply with the Act!

(This contrasts with legislation passed in the U.S. and Canada which *only* applies to the public sector. Other European countries, such as West Germany, France and the Scandinavian countries have enacted laws which apply to both sectors. In some countries, companies as well as individuals are also protected by data laws.)

Perhaps the greatest shortcoming is that the Act deals only with the *automatic* handling of data. While the government is insistent that this is the only area where privacy problems occur (and the European Convention deals only with automatic processing), there are still many manual systems working in the U.K. It would seem that if personal data is collected by hand, and a typewritten copy of it is kept on file, then the data subject has no right of access, and the data user has no obligation to ensure that data is up-to-date and accurate. If data protection is to be taken seriously at all, it must surely cover all forms of data, until such time as there is only one form of data collection and processing.

From a technical point of view, the law does not take into account the present method of storing data. For example, a relational database is one which models the relationships between things as they are in the real world – so an employee may be linked to other employees by salary level, or by geographical location or by number of years service; people seem to be the best item to use as the link in databases. However, if legislation requires personal data to be disclosed to one individual without identifying any other individuals, this will make database design much more difficult.

A further technical difficulty arises when information is given to individuals. In

order that it will be intelligible, data will probably be presented on a printout sheet. However, it would be very easy for a computer user to suppress some personal data so that it is not all printed out – the data subject would not be aware of this at all. Moreover, in order to see data, individuals will have to provide yet more personal data in the form of identification, which is also then open to being recorded.

There is a general lack of technical guidance on how to achieve the objectives of the Act. This is left to the individual data user.

In the short run, the Act will cause great confusion as organisations attempt to decide whether or not they should be registered. Many will probably have to reorganise their data processing systems to improve security and access measures. Most importantly of all, individuals will not know if they are included in files unless they ask. They will not be told automatically.

Data security

As a result of the new Act it will be necessary for computer users to re-evaluate the design of their systems and their security measures. They must ensure that data is correct when it is entered on the computer, and that it remains up-to-date. Unauthorised access to information must also be prevented.

The most obvious step is to tighten up controls over the input, processing and storage of data. One simple exercise, for example, is to **validate** data as it is entered to make sure that the date entry clerk or user has not made any mistakes. Files should be frequently updated, and computing staff should ensure that they are using the 'latest' version of a file in their applications. Regular 'spring cleans' will also help.

Greater physical security can also be applied by the use of security cards etc. Passwords for access to various files can

Science Photo Library/David Parker

Left: a process control module showing displays and oscilloscopes. The operator can monitor all stages of the process.

also be utilised.

However, useful as they are, such measures are not enough. Lists of pass words can be stolen or hit upon by chance. We read of increasing instances of unauthorised entry to computer systems by 'computer hacks' using small personal computers and a little bit of know how. Is there such a thing as a completely foolproof security system?

Science Photo Library/David Parker

Right: this computer console is located in the particle accelerator control room at the CERN laboratory, Switzerland.

Cryptography

Both accidental and deliberate disclosures of information can be prevented by technical measures such as **cryptography** (or **privacy transformation**). Although, theoretically, no code or cipher is unbreakable, the aim of this type of measure is to make the cost and effort involved in breaking the code, outweigh the value of the information it is designed to protect.

The art of cryptography is traditionally associated with radio messages and Morse code, especially during the second world war. However, techniques both of code making and breaking have advanced considerably since the 1940s.

Data transformation or **scrambling** can now be performed very easily by computers, taking place either before the data is stored or transmitted. This is especially important with modern telecommunications techniques where 'electronic snooping' must be prevented.

Messages or passwords can be scrambled and unscrambled using cipher systems. These comprise an algorithm, which is a set of transformations which can be applied to text, and a set of codes or **keys**. The key is used to decide which transformation will scramble the data (encryption) from its normal state (plaintext) to ciphertext. The set of transformations is very large, but is always the same; the set of keys changes.

Some systems use the same key for encryption and decryption, while others use different keys (the Hellman-Diffie method). For example, in **Public Key Algorithms**, a public key is used for encryption while the decryption key is private and secret. If a public directory of personal encryption keys were to be published, then it would be possible to send scrambled messages to anyone which only the intended recipient would be able to unscramble.

The key is vital to the encryption technique – the scrambling and unscrambling keys are mathematically related and it is vital that one part of the key remains secret. It can be thought of as being similar to resettable combination locks – everyone can see the lock, but each user chooses their own number to lock and unlock the door, and no one else should know that number. Outsiders can play with the lock for as long as they like, but the likelihood of them finding the right number is low; the likelihood of them being noticed is much higher.

There are a vast number of encryption transformations which can be used to encrypt a message: the simplest being the most well-known, where different letters are substituted for others in a message, as shown in *figure 2*. This kind of encryption would be very easy to break though, just by examining the statistics of how often letters appear in sentences, like E and S, and which letters can stand alone as one word.

More complicated transformations have therefore been built up, needing computers to produce the encrypted message, and vast amounts of computer power

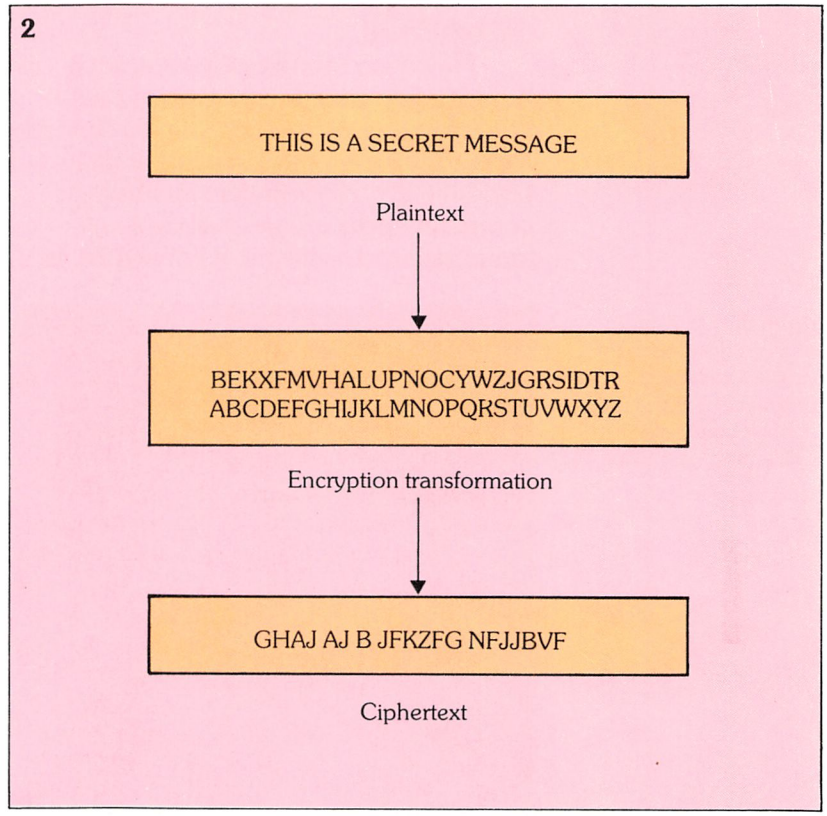
to try to break the cipher.

To make messages even more secure, they can be sent through a number of **nodes**. Here, the message is decrypted and re-encrypted with a new key at each node. This method, called **link encryption**, allows all the data flowing along the link to be encrypted, including the final address. It is necessary that all the users on the network (the nodes) are to be trusted, so this system would work well in, say, a military application, but it might be more difficult to enforce in a commercial organisation.

Another way of increasing security in a network is to incorporate encryption into the password for a remote user. The user logs in at their remote terminal and the host computer sends a message to the terminal which the user has to scramble using their own key. The computer then uses the encrypted messages to check the authenticity of the terminal user.

There is another advantage to data transformation – verifying the authenticity of the message sender. What is needed is the electronic equivalent of a signature, which can only be produced by one person but recognised by many. Suppose that A needs to send a message to B. The message is first encrypted by A with his/her own private decryption (or unscrambling) key. This cipher is then further scrambled using B's public scrambling key. The received message is then unscrambled by B using his public key – but because of the **double scrambling** involved it is still in code. Only by then applying A's public scrambling key can the message be decoded and the identity of the sender verified. The message therefore incorporates the senders own **electronic signature**.

In the U.S., there is now a standard for data encryption, known as the **Data Encryption Algorithm** (DEA). This evolved from a search by the U.S. government, begun in 1973, to find a standard for encrypting non-classified government data. In 1977, a technique submitted by IBM as the Data Encryption Standard became the DEA when it was endorsed by the American standards institute (ANSI). Already, standards have been written for the U.S. banking community, based on DEA. The International Standards Organisation, which deals with most data communications standards, is now developing a worldwide data encryption standard for data processing applications.



2. The simplest type of encryption transformation – substituting one letter for another.

Conclusion

We now have the technology to gather 'information', store it, transfer it around the globe and protect it whilst it is both on the move and in storage. But one of the problems of formulating fair and enforcible data protection laws is that different people attribute a different **value** to the same information.

Another problem is **ownership**. Determining the ownership of details regarding a specific company's product is not too difficult – but who owns personal data regarding an individual's salary, bank balance, credit worthiness or address?

Sometimes information can be considered to be an economic commodity – as in the case of printed information – sometimes it cannot, for example ideas.

The problems of balancing the needs and interests of the individual, industry and commerce, and the state are a long way from being solved.

ELECTRICAL TECHNOLOGY

Active analogue filters

Before looking at other ways of designing low pass filters, we'll first examine a few further types of filter circuit.

To make a **high pass filter**, for instance, a network with series elements that allows current flow at high frequencies, but also acts as an open circuit at low frequencies, is required. This action is provided by a capacitor. As shunt elements in the circuit, components that short circuit at low frequencies, but that have a large impedance at high frequencies are needed – this requirement is satisfied by an inductor. Together, these elements comprise the **high pass Butterworth filter** shown in *figure 1a*. The frequency response of this circuit is given by curve A of *figure 2* – this shows that the gain falls by 60 dB per decade at low frequencies, i.e. the filter is of third order.

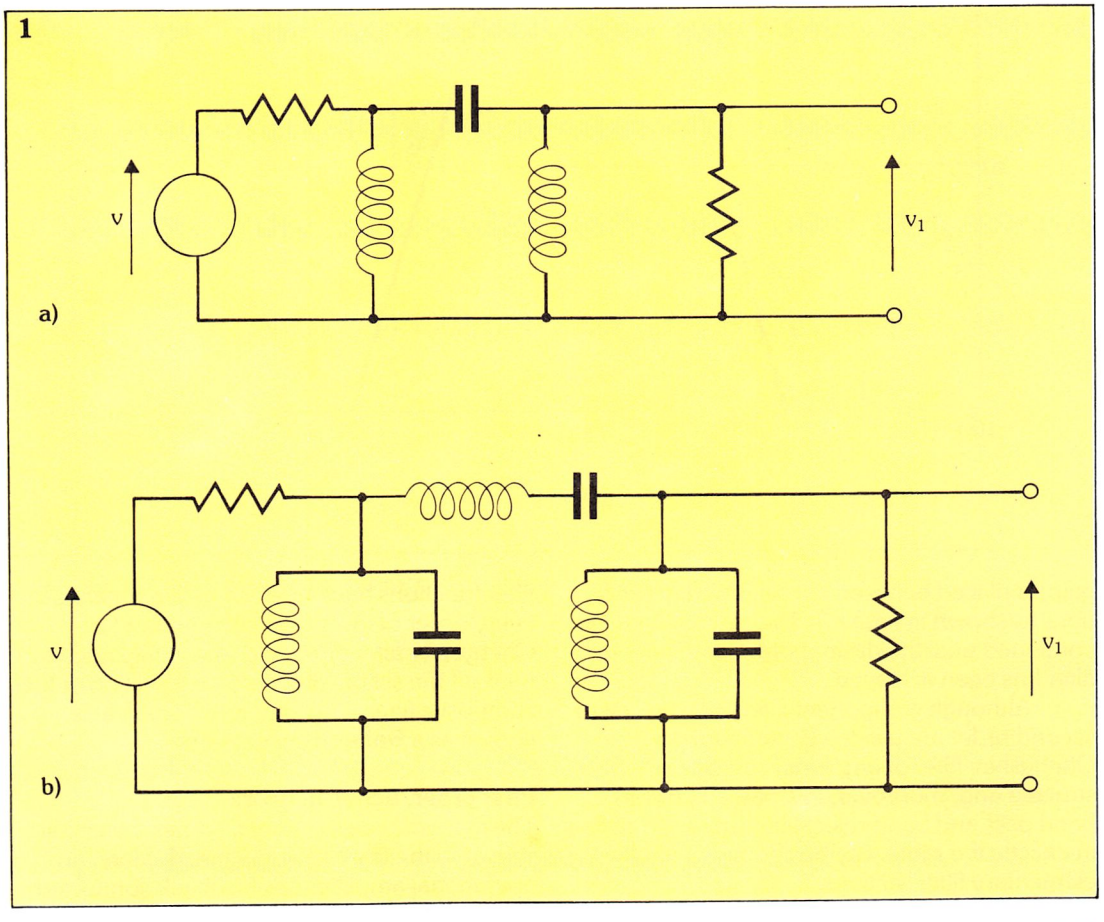
The circuit of a third order **band pass filter** is shown in *figure 1b*. Here, the series elements are series resonant circuits. In an earlier *Basic Theory Refresher*, we found that elements such as this have zero impedance at the resonant frequency, where $f_o = 1/2\pi \sqrt{LC}$; they act like a capacitance at low frequencies and an inductance at high frequencies.

The shunt elements, on the other hand, are parallel resonant circuits. These have an infinitely large impedance at the resonant frequency which falls to low values at other frequences. This kind of filter allows signals in the region of the resonant frequency to pass, but attenuates signals at higher or lower frequencies. The frequency response curve of a third order band pass filter is shown by curve B of *figure 2*.

All filters, whether low pass, high pass, or band pass (even the relatively low orders) use a number of inductors. Discrete inductors consist of coils of wire wound around a core of some magnetic material: high order filters using inductors are therefore large and heavy. If filter networks could be designed without the need for inductors, we would be well on the way to producing a fully integrated circuit form of construction, which is very much smaller.

Active filters

As a first step towards this objective we need to look at the design of **active filters** which are



1. (a) High pass Butterworth filter; (b) third order band pass filter circuit.

constructed only of resistors, capacitors and operational amplifiers, but which have identical properties to those of the passive filters considered above.

Any high order filter may be broken up into a number of lower order filters which may then be connected in cascade. So that individual component filters do not interact, a buffer amplifier (frequently an operational amplifier with feedback connected to give unity gain) is placed between consecutive filter sections as shown in *figure 3*. The cascade may be continued until the desired order of the overall filter has been achieved.

Although component filters of only first or second order are used, a Butterworth or Chebyshev filter of *any* order can be constructed and, moreover, low pass, high pass, band pass and band reject filters may be made in exactly the same way by choosing suitable elementary filter sections.

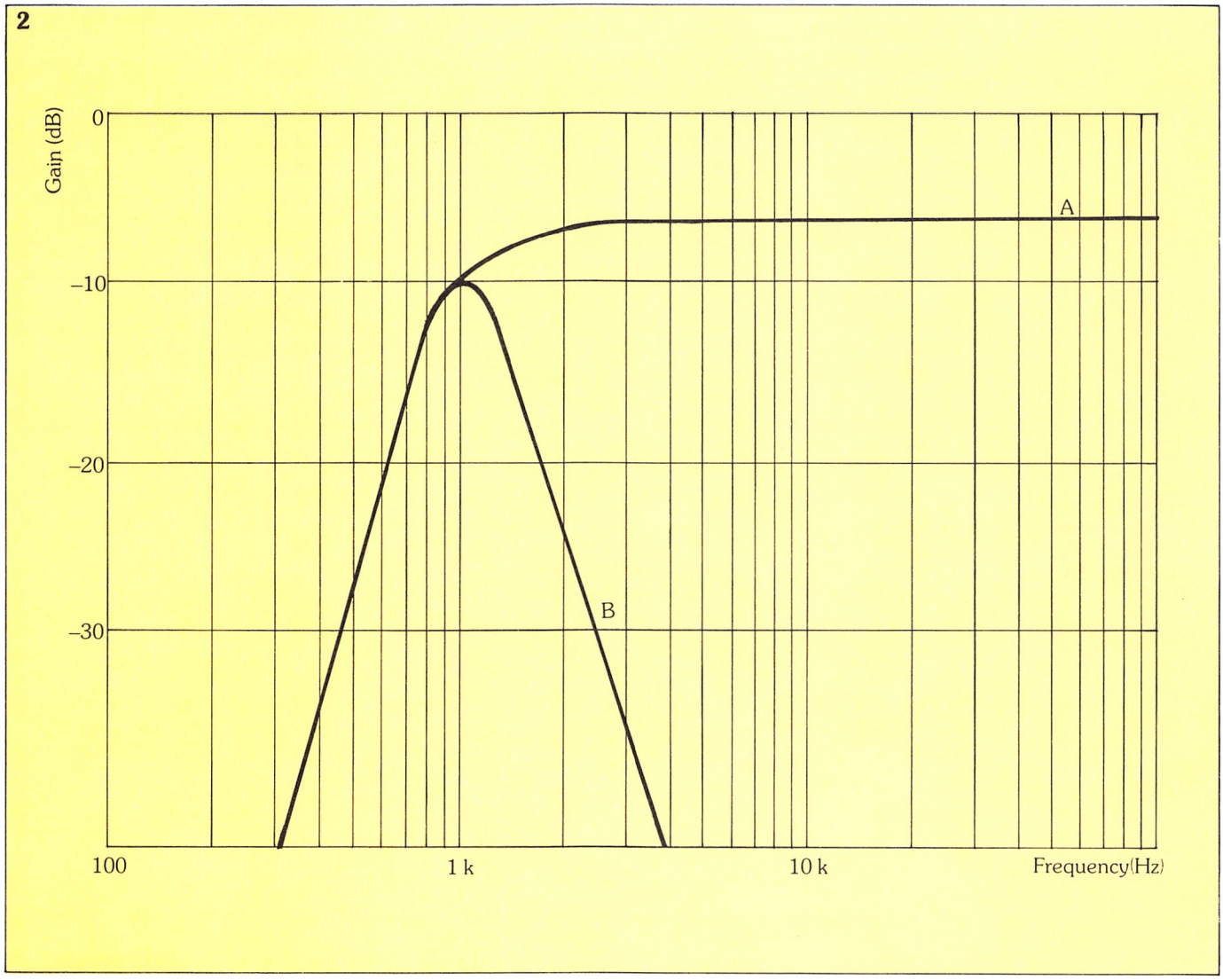
Fifth order, low pass filter

As an example, we'll consider the design of a low pass fifth order Butterworth filter with a cut-off frequency of 20 kHz. This may be designed as a cascade of a first order low pass filter, together with two, second order low pass filters. However, these three elementary filters are *not necessarily Butterworth filters themselves.*

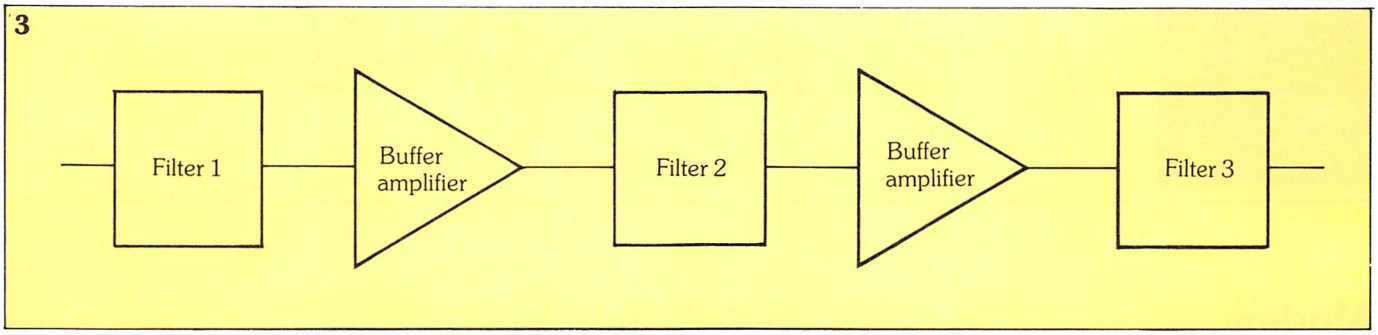
The design is not complicated as tables of filter functions have been compiled for almost every order of low pass Butterworth and Chebyshev filter type, and so we only need to read off the set of values for each section of the composite filter. The filter we will look at is known as a **Sallen-Key network**.

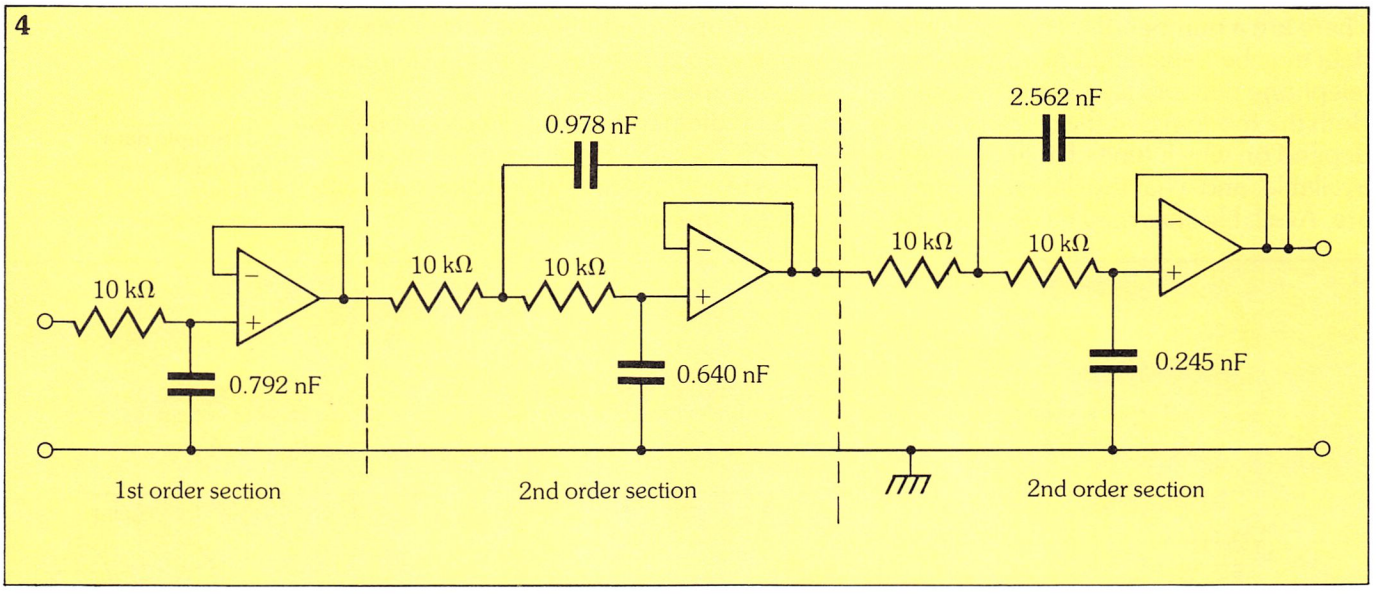
First order, active network

The first order elementary filter has the circuit shown in the first block of *figure 4*. Here, an operational amplifier has feedback connected



2. Frequency response curves of the high pass Butterworth filter (curve A) and the band pass filter (curve B) shown in *figure 1*.





3. An active filter network.

4. A Sallen-Key filter network.

directly from the output to the inverting input (negative feedback), so that the amplifier has a gain of 1 between non-inverting input and output. The resistor and capacitor at the input form a network with a frequency response such that the gain falls at frequencies above the cut-off frequency with a slope of 20 dB per decade. The cut-off frequency, f_o, for this network is given by:

$$f_o = \frac{1}{2\pi RC}$$

$$= \frac{1}{2 \times \pi \times 10^4 \times 0.792 \times 10^{-9}}$$

$$= 20\ \text{kHz}$$

Second order, active network

The two second order networks of *figure 4* are similar in structure although their numerical values differ. Each comprises two operational amplifiers arranged with feedback giving unity gain. The resistors and capacitors are chosen to give the second order behaviour of the filter.

The cut-off frequency of such a second order filter is given by:

$$f_o = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$$

For the first network this gives:

$$f_o = \frac{1}{2\pi\sqrt{10^4 \times 10^4 \times 0.978 \times 10^{-9} \times 0.640 \times 10^{-9}}}$$

$$= 20\ \text{kHz}$$

Similarly, for the second of these networks, f_o is also found to be 20 kHz. As the values of the capacitors in the two networks differ, the network response curves will also be slightly different in the vicinity of the cut-off frequency.

The combination of these three networks may be shown to have a frequencyc response exactly the same as a third order Butterworth filter. Also, the buffer amplifiers shown between the elementary filters in *figure 3* are not actually required as the operational amplifiers provide the necessary isolation.

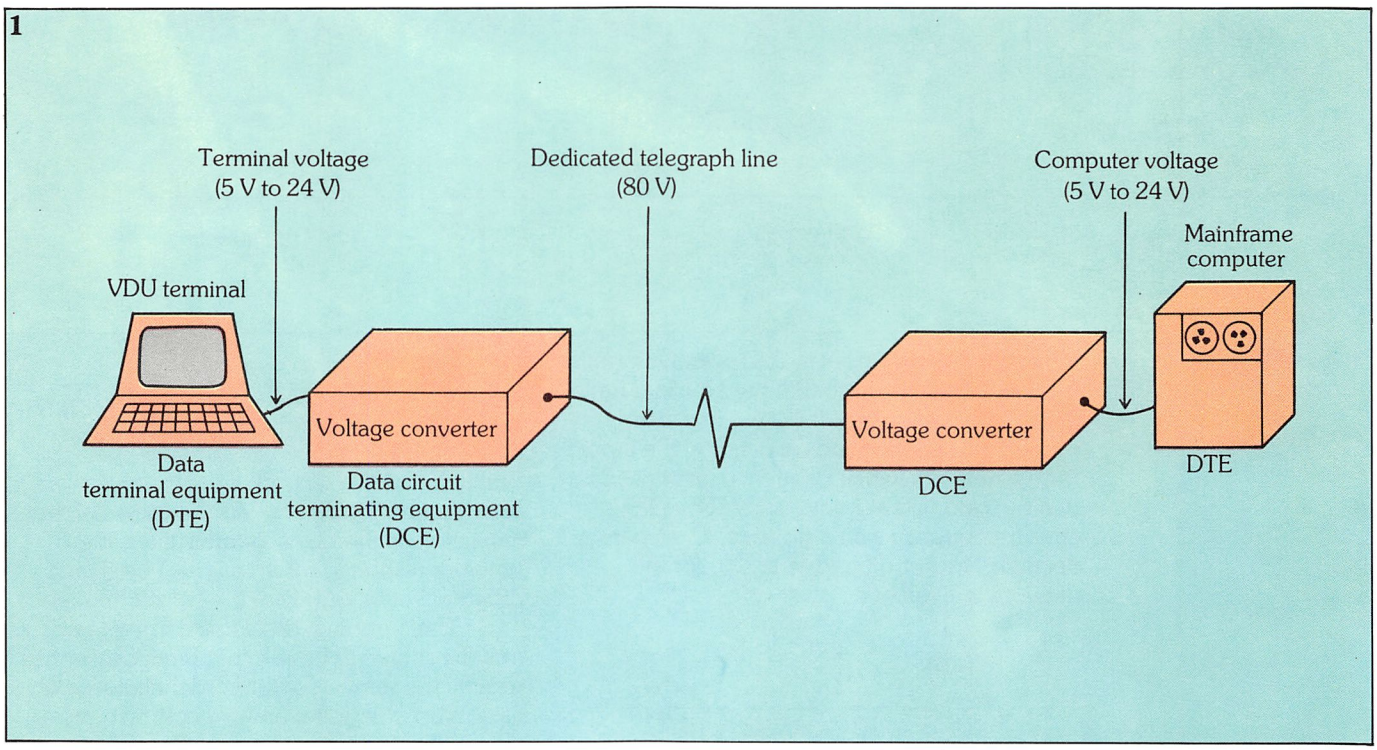
Although we have concentrated on low pass filters, similar design techniques may be used for both high pass and band pass filters. We saw that, in passive networks, a high pass filter could be obtained from a low pass filter by interchanging inductors and capacitors. Similarly, in active networks, a high pass filter is obtained by replacing all resistors and capacitors in a low pass filter by capacitors and resistors, respectively. Similar, more complex transformations may be used to design bandpass filters. □

Data transmission on the PSTN-2

Modems

There are a number of methods by which data may be transmitted over analogue telephone network lines using modems. Both the modem and the method chosen depend on which type of analogue line is available, and what the data requirements are. Available lines may be classified into four main categories:

1. A **telegraph-type circuit**, allowing data transmission speeds of up to 110 bits s^{-1} to be achieved.
2. A **standard telephone connection**, dialled up through the PSTN, with a theoretical capacity of 20,640 bits s^{-1} – this capacity, though, is generally unobtainable.
3. A **dedicated voice grade circuit**, from point-to-point through the PSTN, i.e. a permanent connection between the two modems. **Dedicated lines** (also known as **leased lines**) offer a higher capacity than dialled-up circuits because their signal-to-noise ratio is generally higher. Different grades are available.
4. A **dedicated high-speed circuit** utilising the PSTN's 48 kHz group band circuits. Data rates between 40.8 kbits s^{-1} and 50 kbits s^{-1} are obtainable.



1. Simple data transmission circuit using a voltage converter.

We will now go on to consider a number of different modems which match these categories of analogue line.

Timesharing terminal modes

Terminals such as teletypwriters and VDUs, connected to a computer in timesharing mode, do not require high data capacity because of the low typing and reading speed of humans. Old, slow, teletypewriters such as those used for telex transmissions operate at data signalling

rates of only 50 bits s^{-1} (this is still faster than anyone can type); modern teletype-writers operate at 110 or 300 bits s^{-1}; and VDU terminals usually operate at 300 or 1200 bits s^{-1}. Even when a VDU is used, it is usually possible to select lower rates. Data capacity rates above this are not required on individual timesharing terminal lines.

Modems for this application tend to be fairly inexpensive. The simplest (used on telegraph-type lines) is not really a modem, but rather a voltage converter – converting the computer terminal signal voltage (between 5 V and 24 V, depending on the equipment) to the telegraph line voltage (80 V). If the equipment is capable of generating an 80 V signal voltage, the voltage converter is not required.

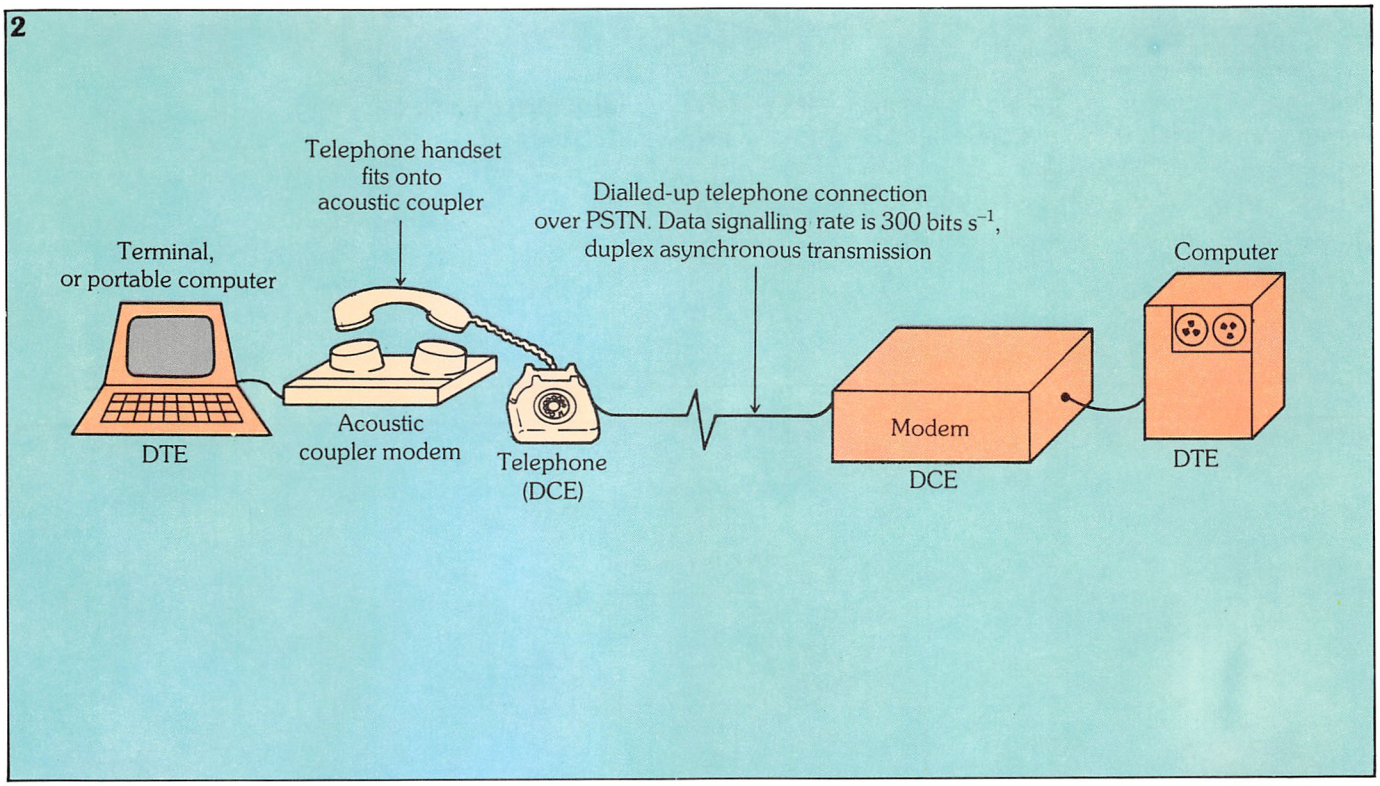
A simple data transmission circuit using a voltage converter is shown in *figure 1*. The terminal and the computer are known as **data terminal equipment** (DTE). As the voltage converters are used to terminate the line, they are known as **data circuit terminating equipment** (DCE). In fact, any equipment providing an interface between the line of a network and any DTE is, by definition, DCE.

Acoustic coupler

Acoustic couplers provide a simple data transmission facility over a dialled-up circuit in the PTSN, without the need for an *electrical* connection to the network. A typical data transmission link using an acoustic coupler, standard telephone terminal and a portable DTE, e.g. a home computer, is shown in *figure 2*.

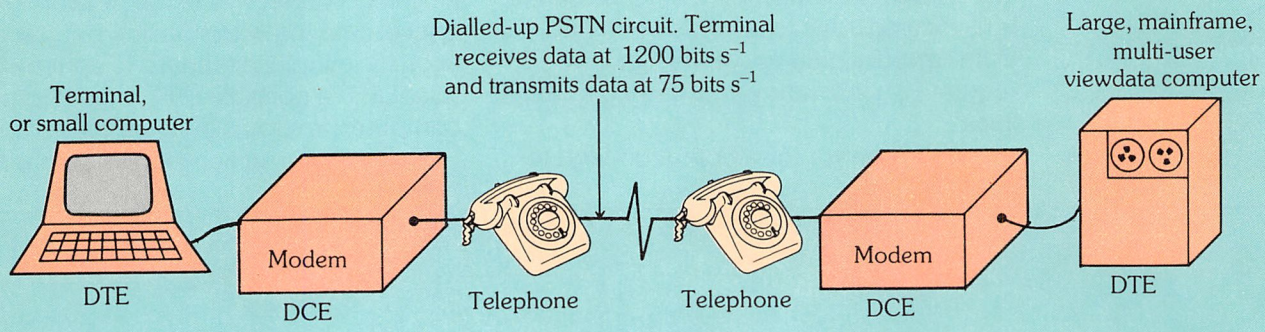
Basically, an acoustic coupler is a modem equipped with a loudspeaker and microphone. The handset of the telephone terminal is fastened, to the acoustic coupler via rubber cups – these help to prevent external background noise from being transmitted. The loudspeaker of the acoustic coupler is thus held close to the mouthpiece of the telephone handset, and its microphone is close to the handset's earpiece. The acoustic coupler then generates tones which are transmitted along the line to the DCE (a *real* modem) and DTE. Data from the DTE is then transmitted by the DCE to the telephone terminal handset earpiece, where the tones are picked up by the acoustic coupler microphone and decoded back to data.

Acoustic couplers are generally used for applications where low data signalling

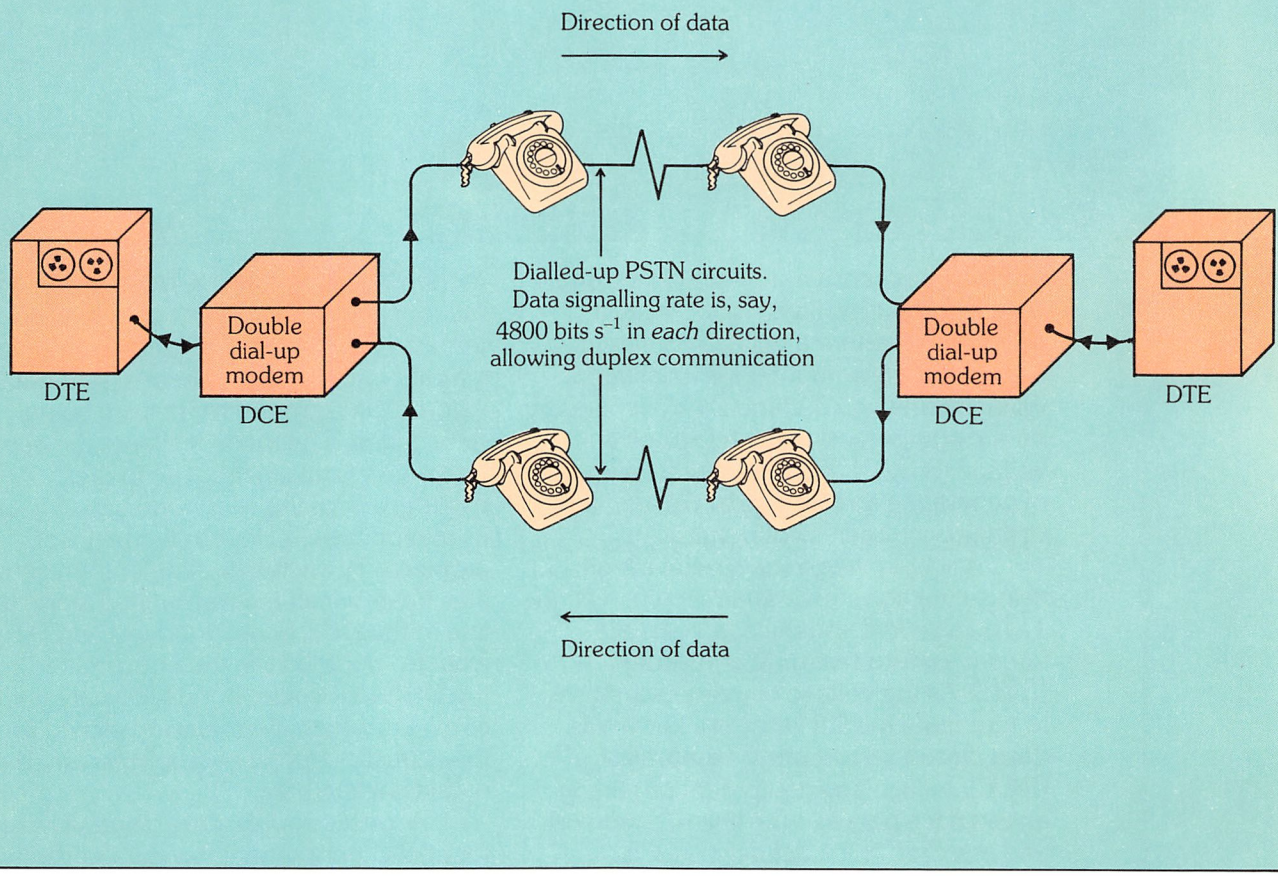


2. Typical data transmission link using an acoustic coupler.

3


Dialled-up PSTN circuit. Terminal receives data at 1200 bits s^{-1} and transmits data at 75 bits s^{-1}
Terminal, or small computer
Large, mainframe, multi-user viewdata computer
Modem
Modem
DTE
DCE
Telephone
Telephone
DCE
DTE

4


Direction of data
Dialled-up PSTN circuits. Data signalling rate is, say, 4800 bits s^{-1} in *each* direction, allowing duplex communication
Double dial-up modem
Double dial-up modem
DTE
DCE
DCE
DTE
Direction of data

3. Connecting a terminal or television receiver to a multi-user computer.

4. Double dial-up modems require two dialled-up connections via the PSTN to enable duplex communications.

5. Split-stream modems enable rates of 19.2 kbits s^{-1} to be achieved.

rates, up to about 300 bits s^{-1}, are required. However, this does not restrict the type of DTE used – the advantage of using any convenient telephone terminal (worldwide) far outweighs the disadvantage of low data rates.

The standard modulation method used with acoustic couplers is frequency shift keying: one pair of frequencies used for data transmission, and another pair for data reception. Duplex communications (according to the CCITT definition) are therefore obtainable. Full modems using this modulation method are also available.

Acoustic couplers and modems of this type only allow asynchronous data transmission, i.e. unclocked.

Viewdata-type modems

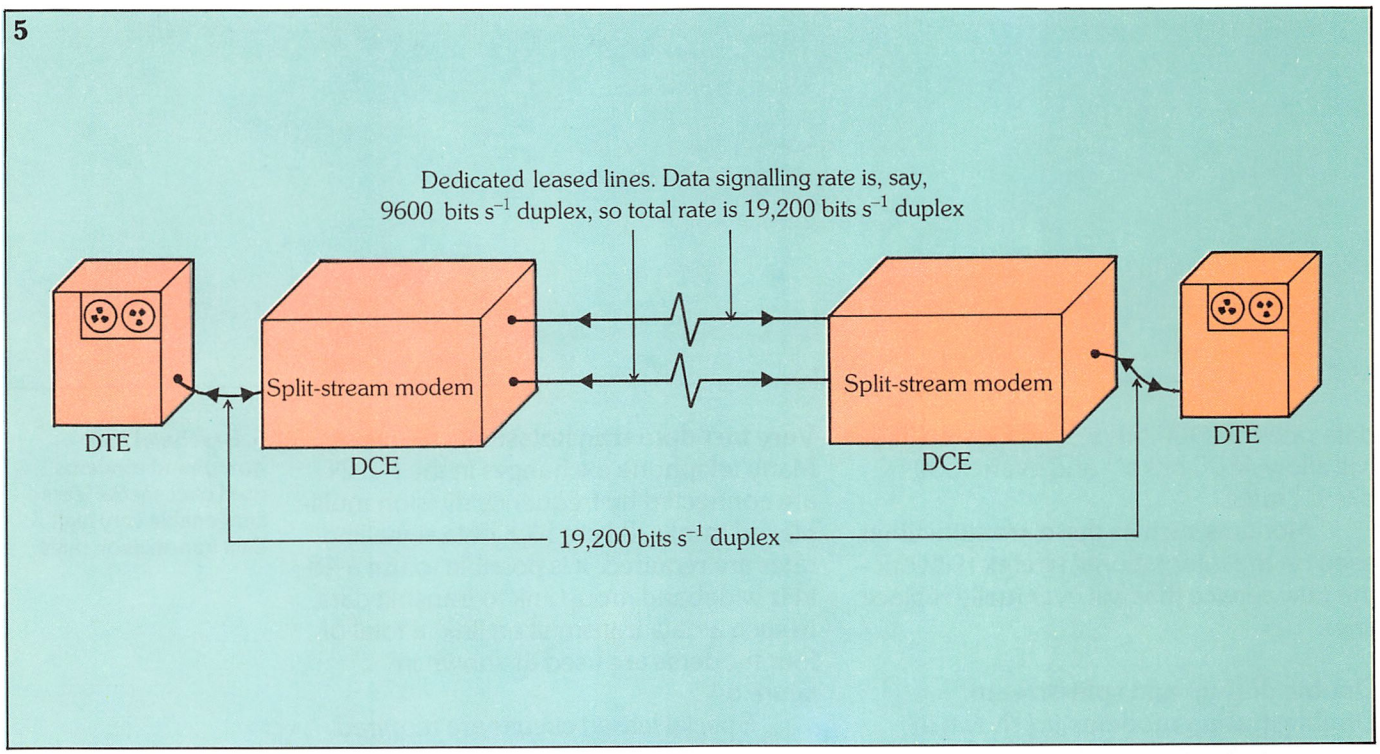
The third type of modem we shall consider is used with **videotex**, i.e. **viewdata** systems, for example BT's Prestel, to connect a terminal or television receiver to a multi-user computer in a half-duplex communications link (CCITT). *Figure 3* illustrates such a link.

These modems enable the terminal DTE to receive data at 1200 bits s^{-1}, but only transmit data at 75 bits s^{-1}.

Although this data rate may seem low, it is perfectly acceptable as videotex is an interactive system and 75 bits s^{-1} allows faster data transmission than a user can enter via a keyboard.

As the terminals used are often home computers, manufacturers build modems to operate the other way round, too (i.e. receiving data at 75 bits s^{-1} and transmitting at 1200 bits s^{-1}). In this way, two home computers can communicate at the higher rate if required. Frequency shift keying is the standard modulation method.

5

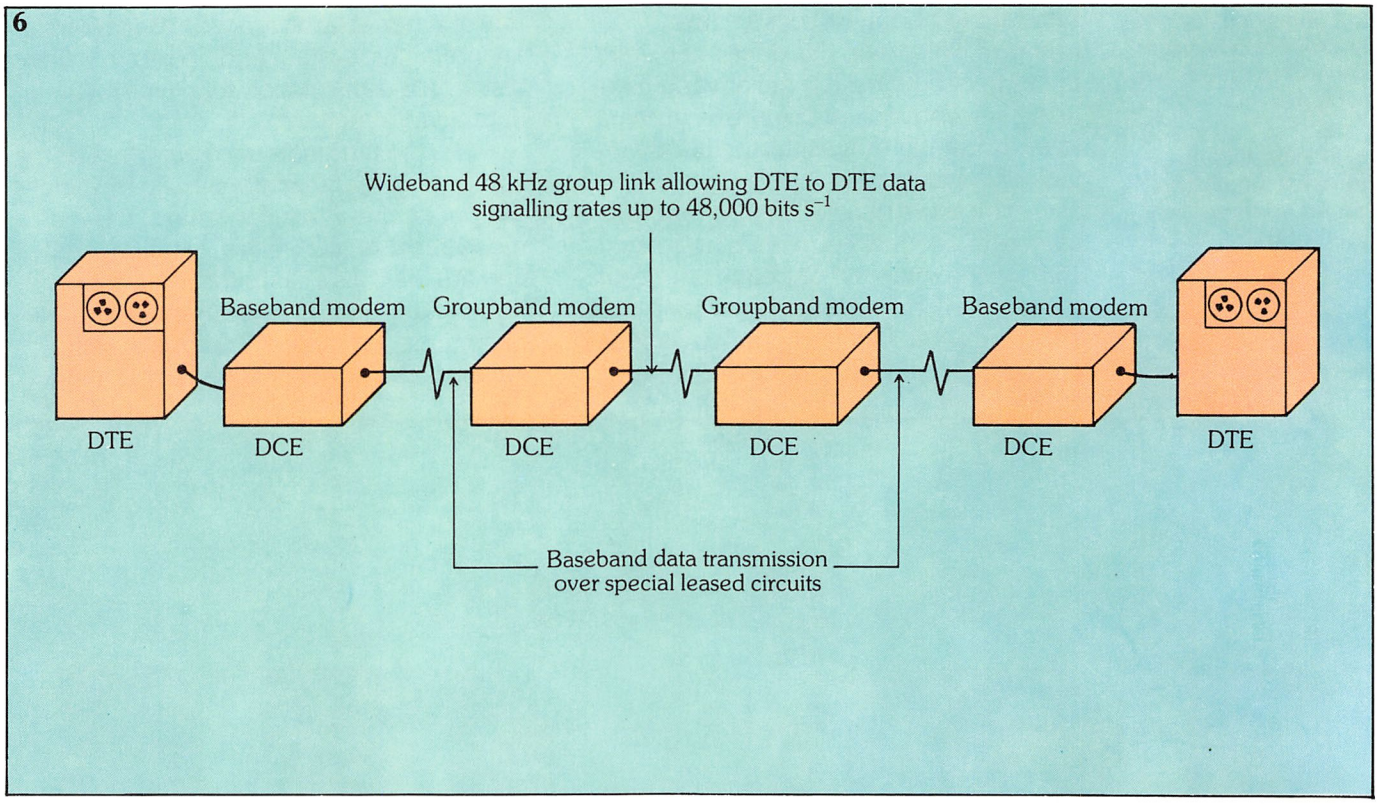


Faster modems

As more complex modulation methods are used in modems, possible data signalling rates correspondingly increase. A duplex (CCITT) 1200 bits s^{-1} modem may be built using a differential four phase modulation method, and used over a dialled-up telephone connection for synchronous or asynchronous data transmission.

Over leased lines such modems may be used with data signalling rates of 2400 bits s^{-1}.

Microprocessor controlled adaptive equalisation (i.e. adjusting the signal received as it alters due to changes in the line's performance) may be used in such modems, enabling even faster data signalling rates. Dialled-up connections allow



6. Baseband and groupband modems used over special leased lines enable very high data transmission rates.

data rates of 2400 bits s^{-1}; and leased lines will allow 4800 bits s^{-1} and even 9600 bits s^{-1} rates.

Modems such as these are sometimes used on the international **teletex** network – the new service that will eventually replace telex.

Double dial-up and split-stream

Double dial-up modems, as shown in *figure 4*, requiring two dialled-up connections via the PSTN are used to allow duplex (CCITT) communications which would otherwise be impossible.

Figure 5 illustrates the use of **split-stream modems**. Two leased lines are used to transmit data in each direction. By transmitting, say, at 9.6 kbits s^{-1} over the parallel lines, a total data signalling rate of 19.2 kbits s^{-1} may be obtained.

Double dial-up and split-stream modem techniques were originally developed in order that higher data signalling rates could be obtained from poor modulation methods. As improved modulation methods are increasingly being used, giving corresponding higher data signalling rates, the use of double dial-up and split-stream techniques is slowly dying out.

Very fast data transmission

Many telephone exchanges in the PTSN are connected by frequency division multiplexed trunks. If very high data signalling rates are required, it is possible to use a 48 kHz wideband group link to transmit data. In such a data transmission link, a total of four modems are used as shown in *figure 6*.

Special leased circuits are required between each DTE and the exchanges which provide the group link. Data is transmitted in these leased circuits as a baseband signal, i.e. a signal containing frequencies down to DC, so a physical pair of wires over a point-to-point connection must be used (the PSTN frequency response is 300 Hz to 3400 Hz, remember). The leased circuits must also be high quality to allow the high data signalling rate.

Modems used to connect DTE to the baseband lines are known as **baseband modems**. Their main task is to modify the binary data signal from the DTE, in a process called **pulse shaping**, by rounding off the square wave pulses to reduce the high frequency components.

(continued in part 44)